NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ANALYSIS OF THE EFFECTS OF PHASE NOISE AND FREQUENCY OFFSET IN ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (OFDM) SYSTEMS

by

Ahmet Yasin Erdogan

March 2004

Thesis Advisor:	Murali Tummala
Co-Advisor:	Roberto Cristi

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Analysis of the Effects of Phase Noise and Frequency Offset in Orthogonal Frequency Division Multiplexing (OFDM) Systems		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Ahmet Yasin Erdogan			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Orthogonal frequency division multiplexing (OFDM) is being successfully used in numerous applications. It was chosen for IEEE 802.11a wireless local area network (WLAN) standard, and it is being considered for the fourth-generation mobile communication systems. Along with its many attractive features, OFDM has some principal drawbacks. Sensitivity to frequency errors is the most dominant of these drawbacks. In this thesis, the frequency offset and phase noise effects on OFDM based communication systems are investigated under a variety of channel conditions covering both indoor and outdoor environments. The simulation performance results of the OFDM system for these channels are presented.

14. SUBJECT TERMS. OFDM (Orthogonal Frequency Division Multiplexing), Frequency Offset, Phase Noise, Differential Decoding, IFFT (Inverse Fast Fourier Transform), Guard Interval, FFT, MATLAB, Additive White Gaussian Noise (AWGN), Interleaver, Deinterleaver, Mobile Channels, Convolutional Encoding, Viterbi Decoder, Probability of Bit Error, Wiener Process.			**15. NUMBER OF PAGES** 150
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ANALYSIS OF THE EFFECTS OF PHASE NOISE AND FREQUENCY OFFSET IN ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (OFDM) SYSTEMS

Ahmet Yasin Erdogan
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2004

Author: Ahmet Yasin Erdogan

Approved by: Murali Tummala
Thesis Advisor

Roberto Cristi
Co-Advisor

John Powers
Chairman, Department of Electrical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Orthogonal frequency division multiplexing (OFDM) is being successfully used in numerous applications. It was chosen for IEEE 802.11a wireless local area network (WLAN) standard, and it is being considered for the fourth-generation mobile communication systems. Along with its many attractive features, OFDM has some principal drawbacks. Sensitivity to frequency errors is the most dominant of these drawbacks. In this thesis, the frequency offset and phase noise effects on OFDM based communication systems are investigated under a variety of channel conditions covering both indoor and outdoor environments. The simulation performance results of the OFDM system for these channels are presented.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
- **A. OBJECTIVE 1**
- **B. RELATED RESEARCH 2**
- **C. ORGANIZATION OF THE THESIS 2**

II. OFDM-BASED IEEE 802.11A STANDARD 3
- **A. INTRODUCTION TO OFDM 3**
- **B. FUNDAMENTALS OF OFDM 6**
  - **1. Multi-Path (Delay-Spread or Time Dispersion) 6**
  - **2. Orthogonality 9**
  - **3. OFDM Transmitter 10**
    - ***a. Channel Coding 11***
    - ***b. Interleaving 11***
    - ***c. Symbol Mapping 12***
    - ***d. Differential Quadrature Phase Shift Keying (DQPSK) Modulation 13***
    - ***e. IFFT 13***
    - ***f. Guard Interval Addition 14***
    - ***g. Number of Sub-carriers 16***
    - ***h. Symbol Pulse Shaping (Windowing) 17***
    - ***i. RF Modulation 17***
  - **4. OFDM Receiver 17**
    - ***a. Removing Guard Interval and FFT Processing 18***
    - ***b. Decoding and Deinterleaving 18***
    - ***c. Received message and Viterbi Decoding 18***
- **C. SUMMARY 19**

III. FREQUENCY ERRORS IN OFDM AND THEIR EFFECTS 21
- **A. FREQUENCY ERRORS 21**
- **B. CREATING A SIMPLE REPRESENTATION OF AN OFDM SIGNAL TO SIMPLIFY ANALYSIS 23**
- **C. THE EFFECT OF DOPPLER FREQUENCY SHIFT, PHASE NOISE AND OSCILLATOR FREQUENCY OFFSET ON THE OFDM SYSTEMS 23**
  - **1. Effect of Doppler Shift on the Carrier Frequencies, Sub-carriers, Envelope and Symbol Timing 24**
  - **2. Phase Noise, Oscillator Frequency Offset and Their Effects 25**
    - ***a. Phase Noise 27***
    - ***b. Frequency Offset 28***
- **D. SUMMARY 30**

IV. OFDM SYSTEM SIMULATIONS IN MATLAB 31
- **A. SELECTING SYSTEM PARAMETERS 31**

B. SIMULATION METHODOLOGY 32
1. Code Check (Channel Model 0) 34
2. OFDM Performance in an AWGN Channel (Channel Model 1) 35
3. OFDM Performance in Mobile Channel (Channel Model 2) 37
a. Mobile Channels *37*
b. Mobile Channel Simulations: *38*
4. Performance with Frequency offset 43
a. Noise Free Channel *43*
b. AWGN Channel *46*
c. Mobile Channel *47*
5. Performance with Phase Noise 48
a. Noise-Free Channel *48*
b. AWGN Channel *51*
c. Mobile Channel *52*
C. SUMMARY 53

V. CONCLUSION 55
A. SUMMARY OF THE WORK DONE 55
B. SIGNIFICANT RESULTS AND CONCLUSIONS 55
C. SUGGESTIONS FOR FUTURE STUDIES 56

APPENDIX A. MATLAB CODE EXPLANATION 59
A. TRANSMITTER 59
1. CDRCDLFT 59
2. TDA 61
B. CHANNEL 61
1. Phase Noise Simulations 61
2. Channel Model Simulations 62
a. Channel Model 0 ..... *62*
b. Channel Model 1 ..... *62*
c. Mobile Channel Models *63*
3. Frequency Offset Simulations 65
C. RECEIVER 65

APPENDIX B. MATLAB CODE 67

LIST OF REFERENCES 127

INITIAL DISTRIBUTION LIST 131

LIST OF FIGURES

Figure 1. A Multi-Path Scenario 6
Figure 2. Delayed Signals (After Ref. 13.) 6
Figure 3. Representation of a Symbol in A Frequency Selective Channel: (a) Time domain (b) Frequency domain 7
Figure 4. Illustration of ISI (After Ref. 14.) 7
Figure 5. (a) Time Domain Representation, (b) Frequency Domain Representation of a Symbol in Flat Fading Channel. 8
Figure 6. OFDM Splits a Data Stream into *N* Parallel Data Streams (After Ref. 15.) 8
Figure 7. Representation of an OFDM Sub-Carrier: (a) In Time Domain (B) Frequency Domain 9
Figure 8. Sub-Carrier Orthogonality (After Ref. 16.) 10
Figure 9. Schematic Diagram of an OFDM Transmitter 10
Figure 10. Convolutional Encoder with Constraint Length= 7. Each box represents a shift register (After Ref. 11.) 11
Figure 11. 16-QAM and QPSK Constellations (From Ref. 11.) 12
Figure 12. Effects of Guard Interval (After Ref. 16.) 16
Figure 13. Schematic Diagram of an OFDM Receiver 17
Figure 14. OFDM Receiver Front End (After Ref. 3.) 22
Figure 15. Effects of Frequency Offset: Reduced Amplitude and ICI (After Ref. 24.) 25
Figure 16. SNR Degradation Versus - 3-dB Bandwidth of the Phase Noise Spectrum for QPSK. 28
Figure 17. SNR Degradation Versus the Normalized Frequency Offset for QPSK. 29
Figure 18. Matlab Simulations Scheme 31
Figure 19. (a) Transmitted and (b) Received QPSK Constellations After Differential Decoding, from a Channel Model 0 Simulation 34
Figure 20. Effect of AWGN on Received Signal Constellation for $s = 0.02$ (Before Differential Decoding) 36
Figure 21. BER versus E_b/N_o Plot for a QPSK Simulation in AWGN Channel 37
Figure 22. Modeling of Mobile Channels 1 and 2 Using FIR Filters and Delay Elements D. 38
Figure 23. Effect of Mobile Channels on Received Signal Constellations (Before Differential Decoding). (a) Mobile Channel 1 (b) Mobile Channel 2 (c) Mobile Channel 3 (d) Mobile Channel 4. 40
Figure 24. Received Signal Constellation Affected by Mobile Channels (After Differential Decoding). (a) Mobile Channel 1 (b) Mobile Channel 2 (c) Mobile Channel 3 (d) Mobile Channel 4. 41
Figure 25. Effect of Mobile Channels on the Magnitude Plot of Received Symbols. (a) Mobile Channel 1 (b) Mobile Channel 2 (c) Mobile Channel 3 (d) Mobile Channel 4. 42
Figure 26. BER versus E_b/N_o Plots for QPSK Simulation in Mobile Channels. 43

Figure 27. Received Signal Constellation in Channel Model 0 with Frequency Offset of 0.1% of Frequency Spacing (Before Differential Decoding) 45
Figure 28. Received Signal Constellation in Channel Model 0 with Frequency Offset of 0.1% of Frequency Spacing (After Differential Decoding) 45
Figure 29. BER versus E_b/N_o Plots for QPSK in AWGN Channel with Relative Frequency Offset Values from 0% to 0.4% 46
Figure 30. BER versus E_b/N_o Plots for QPSK in Mobile Channel 1 with Relative Frequency Offset Values from 0% to 0.4% 47
Figure 31. Phase Noise Generated as a Wiener Process 49
Figure 32. Received Signal Constellation in Channel Model 0 with $b/R_s = 0.0001$. (Before Differential Decoding) 50
Figure 33. Received Signal Constellation in Channel Model 0 with $b/R_s = 0.0001$. (After Differential Decoding) 50
Figure 34. BER versus E_b/N_o Plots for QPSK in AWGN Channel with Normalized Bandwidth b/R_s Values from 0.0% to 0.28%. 51
Figure 35. BER versus E_b/N_o Plots for QPSK in Mobile Channel 1 with Normalized Bandwidth b/R_s Values from 0.0% to 0.28%. 53
Figure 36. Hierarchical m-file Block Diagram for the Transmitter 61
Figure 37. Phase Noise Creating Block in the m-file Hierarchy 62
Figure 38. Channel Model 0 m-file Hierarchy 62
Figure 39. Mobile Channels m-file Hierarchy 64
Figure 40. Summary of Channel Model m-file Architecture 64
Figure 41. Frequency Offset Block in the m-file Hierarchy 65
Figure 42. Hierarchical m-file Block Diagram for the Receiver 66
Figure 43. General m-file Block Diagram 66

LIST OF TABLES

Table 1. OFDM Timing Related Parameters (From Ref. 11.) 4
Table 2. OFDM Rate Dependent Parameters (From Ref. 11.) 4
Table 3. Major Parameters of OFDM Physical Layer (From Ref. 11.) 5
Table 4. Simulation Steps 33
Table 5. Code Check Simulation Parameters 34
Table 6. AWGN Channel Simulation Parameters 36
Table 7. FIR Filter Coefficients for Mobile Channels 1 and 2 38
Table 8. Mobile Channel, Simulation 1 Parameters 39
Table 9. Frequency Offset in Noise Free Channel Simulation 44
Table 10. Frequency Offset in AWGN Channel Simulation 46
Table 11. Phase Noise in Noise Free Channel Simulation 48
Table 12. Phase Noise in AWGN Channel Simulation 51

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS

ADSL	Asymmetric Digital Subscriber Line
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CDMA	Code Division Multiple Access
DAB	Digital Audio Broadcasting
dB	Decibel
DFT	Discrete Fourier Transform
DSSS	Direct Sequence Spread Spectrum
DVB-T	Digital Video Broadcasting-Terrestrial
FCC	Federal Communications Commission
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
GSTN	General Switched Telephone Network
HDSL	High-Rate Digital Subscriber Line
HDTV-T	High Definition TV-Terrestrial
ICI	Inter Carrier Interference
IDFT	Inverse Discrete Fourier Transform
IF	Intermediate Frequency
IFFT	Inverse Fast Fourier Transform
ISI	Inter Symbol Interference
MAC	Medium Access Control
MC-CDMA	Multi Carrier CDMA
OFDM	Orthogonal Frequency Division Multiplexing
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature PSK
RF	Radio Frequency
TDMA	Time Division Multiple Access
UMTS	Universal Mobile Telecommunications System
VHDSL	Very High-Rate Digital Subscriber Line
WLAN	Wireless Local Area Network

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First of all, I want to thank my girlfriend Guzelyasemin GULAY for being in my life and for her support whenever I needed. I also thank my thesis advisors Prof. Murali Tummala and Prof. Roberto Cristi for their patience in answering my questions and for the enjoyable and encouraging atmosphere they created during my whole thesis process. Finally, I want to thank my family beginning with my father Hasan ERDOGAN, who is my inspiration in life, my mother Yadigar ERDOGAN, the strongest woman in my world and my sister Ilknur ERDOGAN along with her son Mert for being a part of my life.

I dedicate this thesis to my family. I feel really lucky to have these wonderful people in my life.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Due to the fact that the need to send more information increases each day, the military is also in need of solutions to make more information transmission to the battlefield possible. The military is looking wireless ADHOC networks as a solution to this problem and orthogonal frequency division multiplexing (OFDM) is the technique used in ADHOC networks.

OFDM is being successfully used in numerous applications. It was chosen for IEEE 802.11a wireless local area network (WLAN) standard, and it is being considered for the fourth-generation mobile communication systems. Along with its many attractive features, OFDM has some principal drawbacks. Sensitivity to frequency errors is the most dominant of these drawbacks.

The objective of this thesis was to investigate the frequency offset and phase noise effects on OFDM based communication systems to determine the ways by which we can use this technique in battlefield applications more efficiently. A comprehensive background of OFDM based communication systems is presented, and the sub-carrier orthogonality requirement, inverse fast Fourier transform and fast Fourier transform processes are discussed. Theoretical analysis of frequency errors and their effects on the overall system performance are examined based on the work reported in the literature.

Matlab simulations of an OFDM communication system were performed for the following channels: ideal, additive white Gaussian noise (AWGN), and mobile. The mobile channel selected includes multipath and Doppler effects in AWGN. The effects of frequency offset and phase noise were studied for all channel models by observing the performance deviation from the AWGN case as a function of increasing frequency offset and phase noise. The bit error rate (BER) performance plots of the OFDM system affected by frequency errors under AWGN and mobile channel conditions were obtained through simulations.

It was observed that the OFDM system's performance degraded as a power of the relative frequency offset. A relative frequency offset of 0.1% required a bit energy to noise power spectral density ratio (E_b/N_o) value of 1 dB more than the AWGN case without offset, and a frequency offset of 0.2% required 2.8 dB more E_b/N_o for the same BER.

In the phase noise case, it was observed that the OFDM performance is degraded linearly as we increased the normalized bandwidth, as was suggested in the literature. A normalized bandwidth of 0.07%, representing the amount of phase noise, required an E_b/N_o value of 1 dB more than the AWGN case without phase error, and each additional 0.07% increment in the normalized bandwidth resulted in 1 dB more E_b/N_o requirement.

The effects of mobility were also studied. The mobility is simulated by including the Doppler frequency shift and multipath effects. The simulations suggested that the signal to noise ratio degradation due to the Doppler shift, most of the time, is negligible. The degrading effects of multipath are observed by comparing the system performance under mobile channels with that under AWGN conditions.

I. INTRODUCTION

Orthogonal frequency division multiplexing (OFDM) is being successfully used in numerous applications, such as European digital audio broadcasting and digital video broadcasting systems [1, 2]. In 1999, the IEEE 802.11a working group chose OFDM for their 5-GHz band wireless local area network (WLAN) standard, which supports a variable bit rate from 6 to 54 Mbps. OFDM was also one of the promising candidates for the European third-generation personal communications system (universal mobile telecommunication system). However, it was not approved since the code division multiple access (CDMA)-based proposals received more support. OFDM is now being considered for the fourth-generation mobile communication systems [3]. Therefore, OFDM's performance in mobile and fading environments is the topic of many current studies. These studies are also helpful to determine the ways by which we can use this technique in battlefield applications more efficiently.

A. OBJECTIVE

Although OFDM seems to be a solution to keep up with the demand of increasing data rates, it has some drawbacks. Sensitivity to frequency errors is the most significant of these drawbacks. The main objective of this thesis was to investigate and document the effects of frequency offset and phase noise on the performance of OFDM based digital communications under different channel conditions.

A step-by-step approach was adopted in order to achieve the objective of this thesis. The first step is to provide a basic background on the principles of OFDM. The reasons for the frequency errors and a theoretical analysis of these effects on OFDM systems are documented. Finally, a communication system utilizing OFDM was simulated in Matlab and the system's behavior under different channel conditions was observed.

To be able to observe the system behavior, the simulation results for different channel models are presented in graphical form. Next, the simulation results obtained in this work are compared to the simulation results reported in related studies.

B. RELATED RESEARCH

Due to its many attractive features, OFDM has received much attention in the wireless communications research communities. Numerous studies have been performed to investigate its performance and applicability to many different environments. Below are some of the many studies conducted concerning the effect of frequency errors on OFDM systems.

Many of the published studies about the frequency errors use two main references. The first is the study of Pollet *et al*. on sensitivity of OFDM systems to frequency offset and Wiener phase noise [4], and the second is the study of Moose on a technique for OFDM frequency offset correction [5].

Other related studies include the study of Armada on the phase noise and subcarrier spacing effects on OFDM systems' performance [6], the study of Xiong about the effect of Doppler frequency shift, frequency offset, and phase noise on OFDM receivers' performance [7] and the study of Zhao *et al*. on the sensitivity of OFDM systems to Doppler shift and carrier frequency errors [8].

C. ORGANIZATION OF THE THESIS

Chapter II starts with an introduction to OFDM and proceeds to present the fundamentals of OFDM. The generation of an OFDM symbol is also explained using a step-by-step approach. Chapter III discusses the frequency errors including the frequency offset and the phase noise. A simple OFDM signal representation is adopted to easily represent the frequency offset and Doppler frequency shift effect in analytical expressions. The reasons for frequency errors are explained.

Chapter IV develops the Matlab simulations to study the system performance under different environments. It also presents the results. Chapter V provides a summary of the thesis, the conclusions and suggestions for future work.

Appendix A outlines the Matlab code for the transmitter, the channel and the receiver. The function of each m-file and its relationship to other functions are discussed. At the end of each section, the hierarchy of the m-files is illustrated in a schematic form. Appendix B presents the Matlab code.

II. OFDM-BASED IEEE 802.11A STANDARD

The rapid growth of the applications utilizing digital communication systems increased the need for high-speed data transmission. New multi-carrier modulation techniques are being proposed and implemented to keep up with the demand of higher data rates. Of these multi-carrier techniques, OFDM is the method of choice for high-speed communication due to its many attractive features. This chapter attempts to justify the choice of OFDM among other communication techniques.

A. INTRODUCTION TO OFDM

OFDM was introduced in the 1950s but was first implemented in the 1960s. It was originally developed from the multi-carrier modulation techniques used in high frequency military radios. A patent for OFDM was granted in the 1970s. However, when OFDM was first introduced, it was not very popular because of the cost and complexity of large arrays of sinusoidal generators and coherence demodulators [9].

The actual widespread use of OFDM started after the inverse discrete Fourier transform (IDFT) and discrete Fourier transform (IDFT) made the OFDM implementation possible without the use of large number of sinusoidal generators.

OFDM was accepted as a wireless local area network (WLAN) standard in September 1999, but actually it was not the first IEEE physical standard for WLANs. The first standard was approved in June 1997 and specified one medium access control (MAC) and three physical layers (IEEE 802.11 FHSS, IEEE 802.11 DSSS and IEEE 802.11 IR). The IEEE 802.11 direct sequence spread spectrum (DSSS) originally supported both 1 Mbps and 2 Mbps of data rates while the other two supported 1 Mbps or 2 Mbps optionally. Due to the increasing demand for higher throughput, a high-data-rate DSSS (IEEE 802.11b) was selected for standardization in July 1998, and the data rate increased to 11 Mbps. The IEEE 802.11a and IEEE 802.11b standards were developed simultaneously. The federal communications commission (FCC) released 300 MHz of spectrum in the 5.2-GHz band in January 1997 for WLAN applications, and IEEE 802.11a was targeted to use this spectral band [10].

OFDM was chosen to be the physical standard for IEEE 802.11a. The time-related parameters of OFDM for IEEE 802.11a are listed in Table 1.

Parameter	**Value**
Number of data sub-carriers	48
Number of pilot sub-carriers	4
Total number of sub-carriers	52
Sub-carrier frequency spacing	0.3125 MHz
IFFT/FFT period	$3.2\,\mu s\ \ (1/\Delta_F)$
Preamble duration	$16\,\mu s$
Signal Duration BPSK-OFDM symbol	$4\,\mu s\ \ (T_{GI}+T_{FFT})$
Guard Interval (GI) duration	$0.8\,\mu s\ \ (T_{FFT}/4)$
Training symbol GI duration	$1.6\,\mu s\ \ (T_{FFT}/2)$
Symbol interval	$4\,\mu s\ \ (T_{GI}+T_{FFT})$
Short training sequence duration	$8\,\mu s\ \ (10\,T_{FFT}/4)$
Long training sequence duration	$8\,\mu s\ \ (T_{GI}+2T_{FFT})$

Table 1. OFDM Timing Related Parameters (From Ref. 11.)

The rate dependent parameters of IEEE 802.11a are listed in Table 2. As can be observed from Table 2, different sub-carrier modulation schemes can be used in combination with different convolutional codes to achieve the desired data rate for the application.

Data rate (Mbps)	**Modulation**	**Coding Rate(R)**	**Coded bits per sub-carrier (N_{BPSC})**	**Coded bits per OFDM symbol (N_{CBPS})**	**Data bits per OFDM symbol (N_{DBPS})**
6	BPSK	1/2	1	48	24
9	BPSK	3/4	1	48	36
12	QPSK	1/2	2	96	48
18	QPSK	3/4	2	96	72
24	16-QAM	1/2	4	192	96
36	16-QAM	3/4	4	192	144
48	64-QAM	2/3	6	288	192
54	64-QAM	3/4	6	288	216

Table 2. OFDM Rate Dependent Parameters (From Ref. 11.)

Table 3 shows the major parameters of the OFDM physical layer. In the simulations conducted in this thesis, a data rate of 24 Mbps, modulation type of QPSK, and a coding rate of ½ were used, while the other parameters in Table 3 were left unchanged.

Information data rate	6,9,12,18,24,36,48 and 54 Mbps (6,12 and 24 Mbps are mandatory)
Modulation	BPSK OFDM QPSK OFDM 16-QAM OFDM 64-QAM OFDM
Error correcting code	$K = 7$ convolutional code
Coding rate	1/2, 2/3, 3/4
Number of sub-carriers	52
OFDM symbol duration	4 μs
Guard interval	0.8 μs (T_{GI})
Occupied bandwidth	16.6 MHz

Table 3. Major Parameters of OFDM Physical Layer (From Ref. 11.)

OFDM is being used in a number of wired and wireless voice and data applications due to its flexible system architecture. Some examples of current OFDM applications are DAB (digital audio broadcasting), HDSL (High-Rate Digital Subscriber Line), VHDSL (Very High-Rate Digital Subscriber Line), ADSL (Asymmetric DSL), HDTV-Terrestrial (High Definition TV-T), IEEE 802.11 and HiperLAN/2, GSTN (General Switched Telephone Network), Cellular Radio and DVB-T (digital video broadcasting-terrestrial) [9].

From the range of the current applications and the research activity on OFDM and considering the fact that many wireless communication systems and network standards are being developed based on OFDM, it may be easily remarked that this technique will have a significant impact on the future of digital communications and wireless networks.

B. FUNDAMENTALS OF OFDM

1. Multi-Path (Delay-Spread or Time Dispersion)

In general, high data rate means short symbol time compared to the delay spread $\left(T_{symbol} < T_{delay}\right)$. Delay-spread greatly affects the communication system and the signal might not be recovered at the receiver.

This section addresses the effects of delay spread which occurs as the surfaces between a transmitter and a receiver reflect a transmitted signal as in Figure 1.

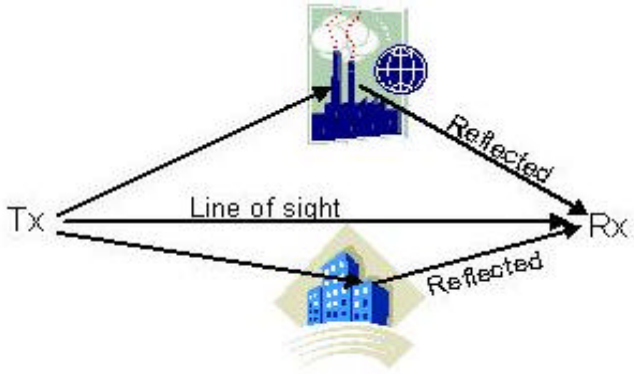


Figure 1. A Multi-Path Scenario

The receiver obtains the transmitted signals with random phase offsets and this causes random signal fades as reflected signals destructively or constructively affect each other [12], as seen in Figure 2.

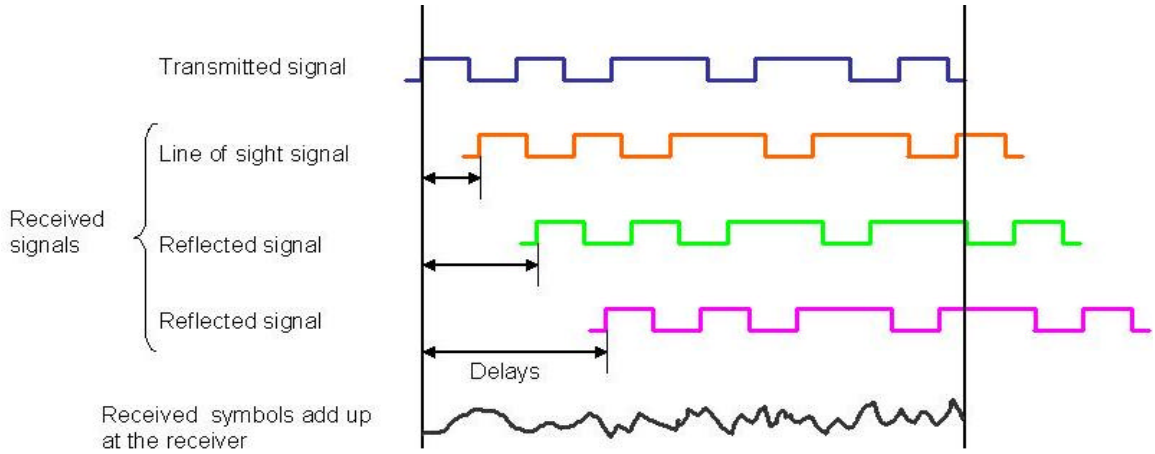
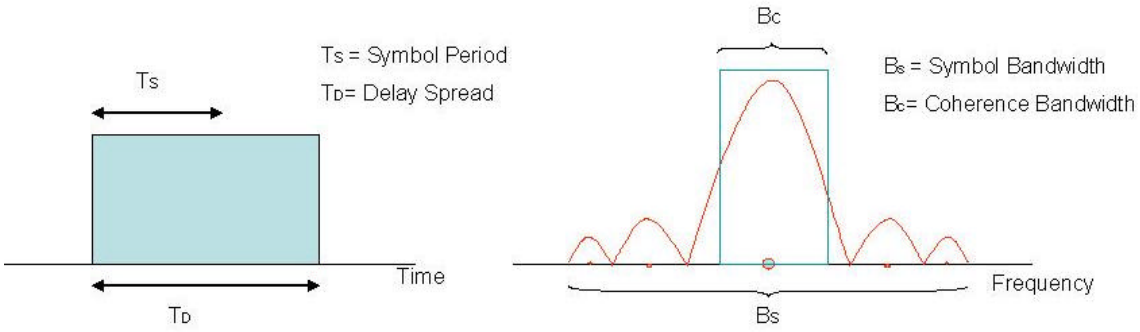


Figure 2. Delayed Signals (After Ref. 13.)

When $T_{symbol} < T_{delay}$ $\left(B_c < B_s\right)$, as in Figure 3, the signal faces frequency selective fading and this causes time dispersion. The effect of this is intersymbol interference (ISI), where the energy of one symbol leaks into another symbol, as can be viewed from Figure 4. As a result, the bit error rate (BER) increases, this in turn degrades the performance. ISI is one of the biggest problems of digital communication and OFDM deals with this problem very effectively.



(a) (b)

Figure 3. Representation of a Symbol in A Frequency Selective Channel: (a) Time domain (b) Frequency domain

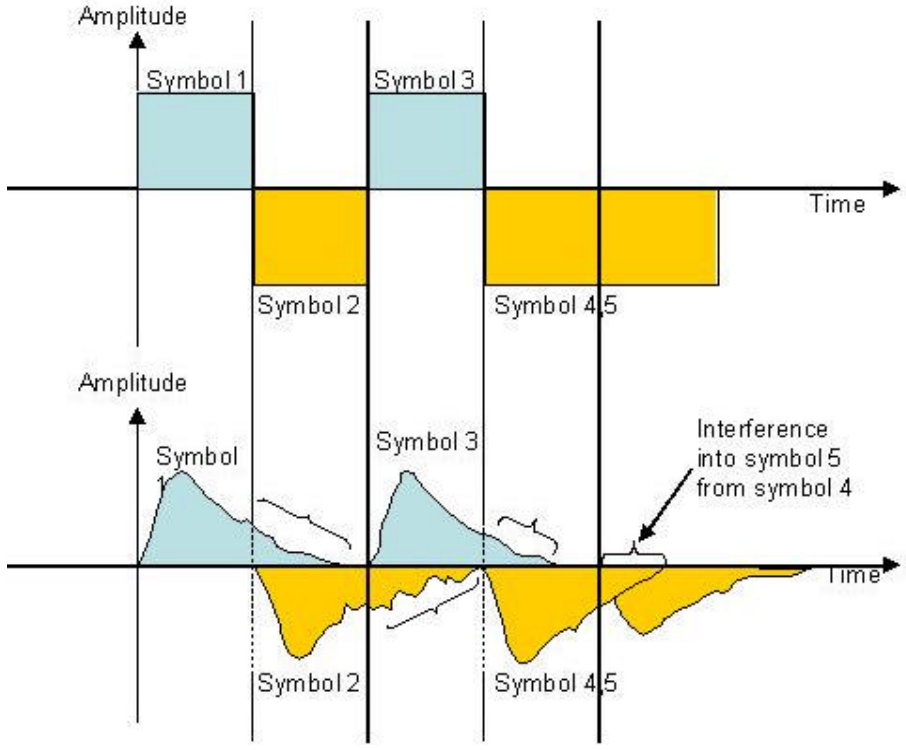


Figure 4. Illustration of ISI (After Ref. 14.)

A way to deal with frequency selective fading is to decrease the data rate and thus change the frequency selective fading to flat fading. The desired scheme is illustrated in Figure 5. OFDM systems mitigate the ISI by changing the frequency selective fading channel to flat fading channel as discussed below.

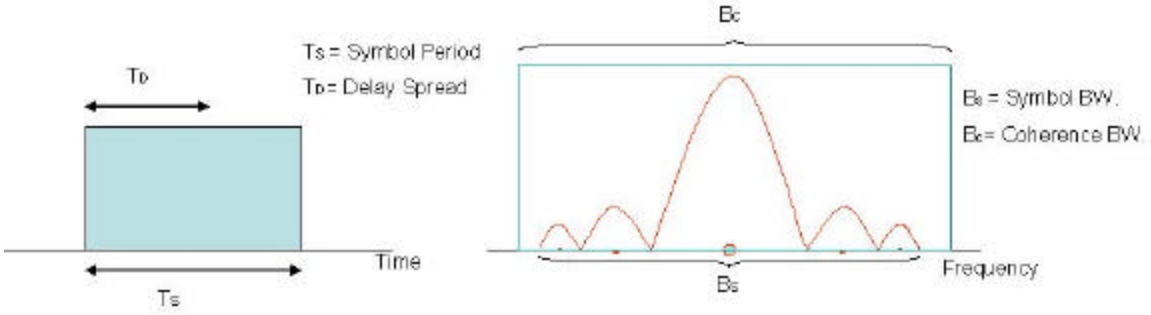


Figure 5. (a) Time Domain Representation, (b) Frequency Domain Representation of a Symbol in Flat Fading Channel.

OFDM modulates user data onto tones by using either phase shift keying (PSK) or quadrature amplitude modulation (QAM). An OFDM system takes a high data rate stream, splits it into *N* parallel data streams and transmits them simultaneously. As can be observed from Figure 6, each of these parallel data streams has a rate of R/N, where *R* is the original data rate. The data streams are modulated by different carriers and combined together by inverse fast Fourier transform (IFFT) to generate the time-domain signal to be transmitted [12].

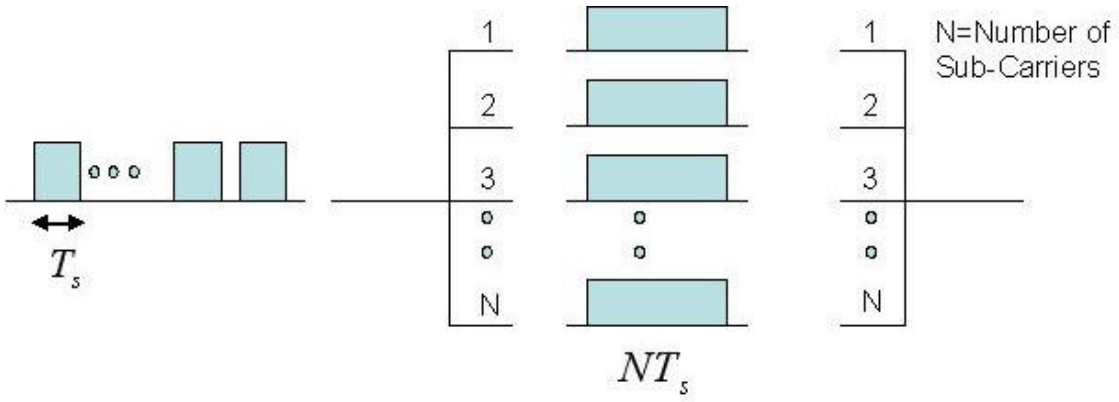


Figure 6. OFDM Splits a Data Stream into *N* Parallel Data Streams (After Ref. 15.)

By creating a slower data stream, the symbol duration becomes larger than the channel's impulse response. In this way, each carrier is subject to flat fading.

2. Orthogonality

Normally, multicarrier systems, such as frequency division multiplexing (FDM), have to modulate different sub-carriers with spectrally separate symbols to avoid inter carrier interference (ICI) at the cost of a bandwidth loss. However, in OFDM, spectrally overlapped sub-carriers can be used and since they are orthogonal, they do not interfere with each other. This makes OFDM a bandwidth efficient modulation scheme [12].

Orthogonality of the sub-carriers must be ensured to avoid ICI. OFDM carriers are orthogonal over a symbol interval only if they are spaced in frequency exactly at the inverse of the symbol period (see Figures 7 and 8). IFFT implementation inherently provides this frequency spacing requirement. It is also necessary to add a guard interval in multi-path channels to account for the multi-path spreading effects. Figure 7 shows a sub-carrier having a rectangular pulse shape and its frequency spectrum. The rectangular pulse has a width of T_s and has a sinc-spectrum with nulls at multiples of $1/T_s$.

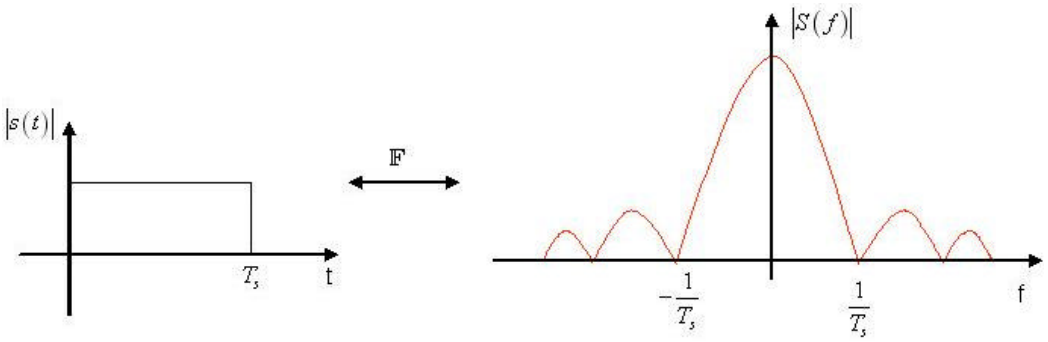


Figure 7. Representation of an OFDM Sub-Carrier: (a) In Time Domain (B) Frequency Domain

Figure 8 shows the spectra of multiple carriers. Although all frequency spectra are overlapping, they do not overlap at the carrier frequencies $f_1, f_2, \ldots$, and this implies orthogonality. In the time domain each carrier has an integer number of periods within the symbol time interval. Due to the orthogonality, the nulls of the side lobes coincide with

the peaks of the main-lobes of each carrier. By building the receiver to sample at the center frequency of each sub-carrier, it is possible to obtain only the corresponding signal's energy, since the side-lobes of other carriers have zero energy at that frequency [12].

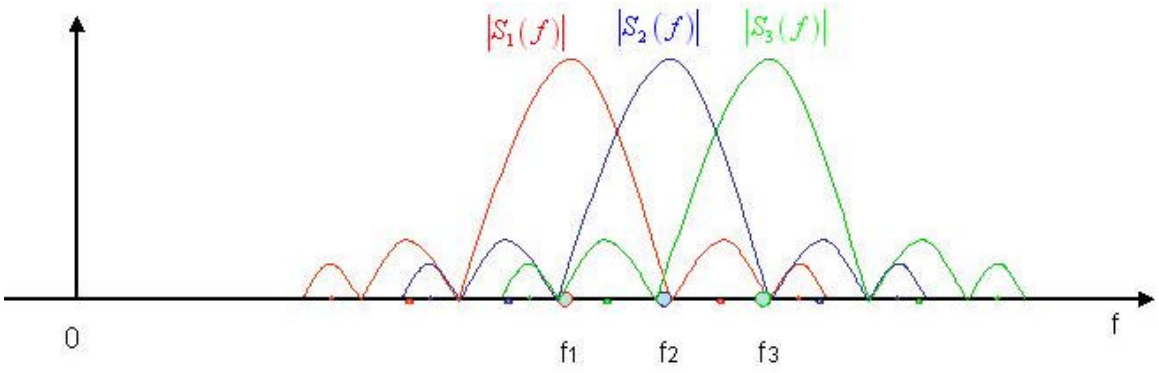


Figure 8. Sub-Carrier Orthogonality (After Ref. 16.)

Orthogonality can be lost due to two main reasons. The first reason is the frequency errors, which are discussed in detail in Chapter III, and the second is the error in symbol boundary and sampling rate offset.

3. OFDM Transmitter

A block diagram of the OFDM transmitter module is presented in Figure 9. Each of the blocks is explained in detail in the following subsections.

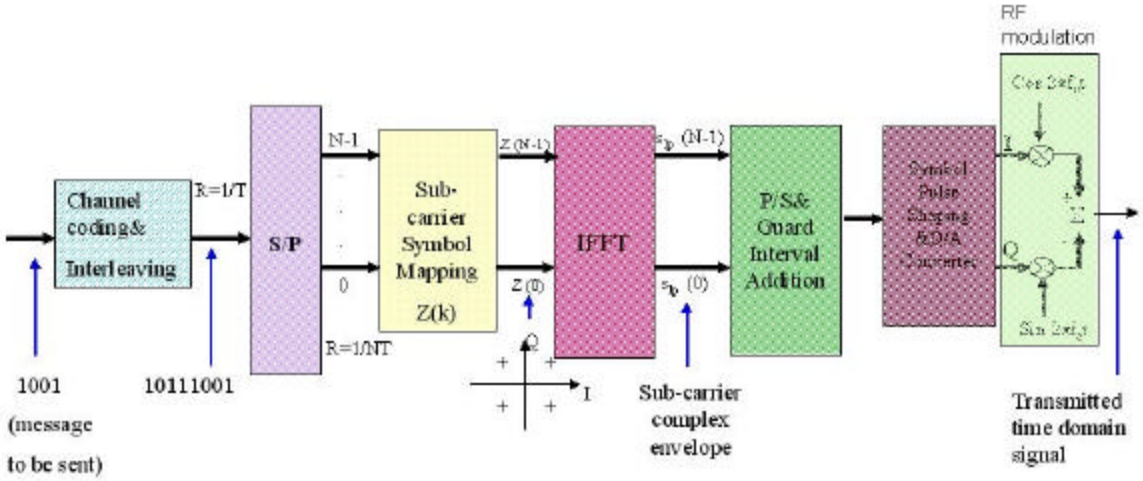


Figure 9. Schematic Diagram of an OFDM Transmitter

a. Channel Coding

A sequential binary input data stream is first encoded by the channel coder. Error correction coding is important for OFDM systems used for mobile communications. When channel coding is used to improve its performance, OFDM is referred to as coded OFDM (COFDM). The IEEE 802.11a standard uses convolutional coding as forward error correction (FEC) coding, while the convolutional coder has rate ½, constraint length 7 and generator polynomials (133, 171) [11].

Table 2 lists the rate and modulation scheme combinations for a desired value of data rate. Higher data rates are obtained by puncturing the rate ½ code. The Convolutional encoder used in the IEEE 802.11a standard is presented in Figure 10.

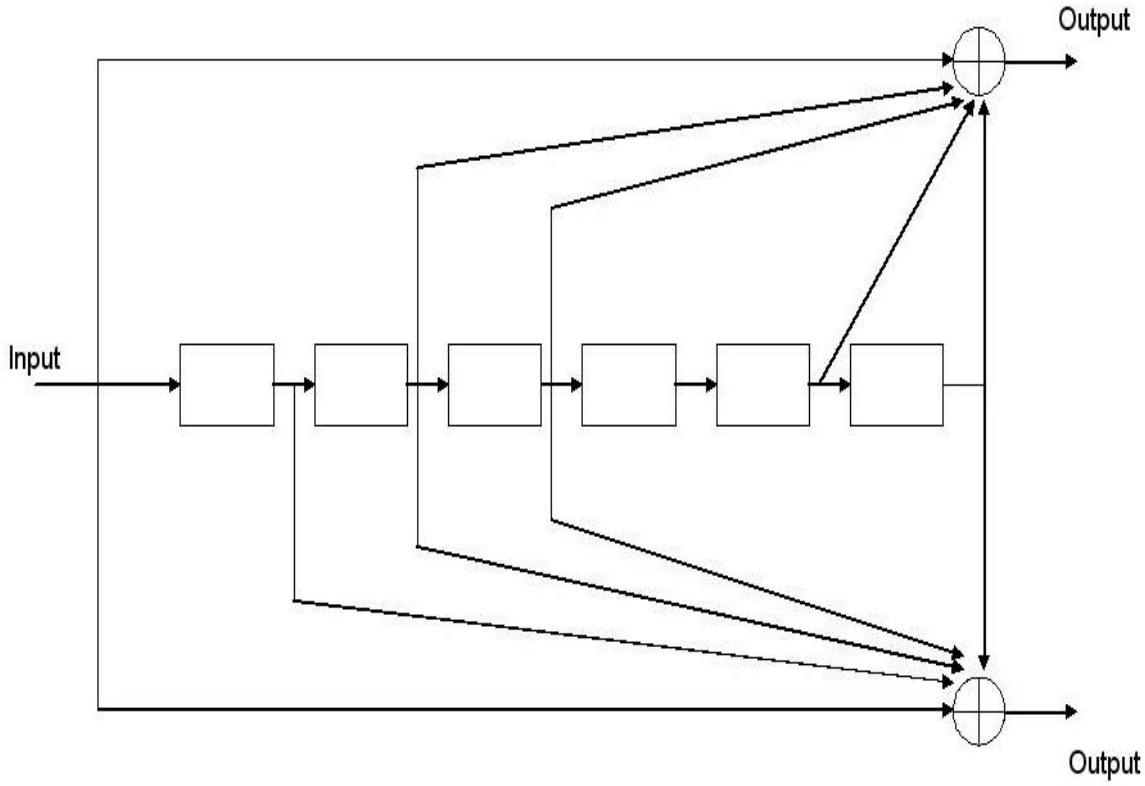


Figure 10. Convolutional Encoder with Constraint Length= 7. Each box represents a shift register (After Ref. 11.)

b. Interleaving

The encoded symbols, out of the convolutional encoder, are next interleaved by a block interleaver. The block interleaver takes data from the encoder, per-

mutes it and then outputs the rearranged binary data. The idea behind interleaving is to handle burst errors. The interleaver simply scatters the burst errors randomly so that the receiver can easily correct the random errors.

Many interleaving methods exist, but the conventional block interleaver is widely used since it is the easiest to implement. To implement block interleaving, a matrix that can accommodate all the symbols in the transmit message matrix must first be created. Next, the symbols from the message matrix are read into this intermediate matrix by rows and the interleaving effect is created by reading these symbols out by columns. The degree of interleaving is determined by the dimensions of this intermediate matrix.

c. Symbol Mapping

For the symbol mapping part of the block diagram, the required data rate for the application will determine which signal constellation, i.e., which modulation technique is to be used.

Following the interleaver, the rearranged binary data are mapped onto the desired constellation points. Figure 11 shows two constellation examples that were listed in Table 2.

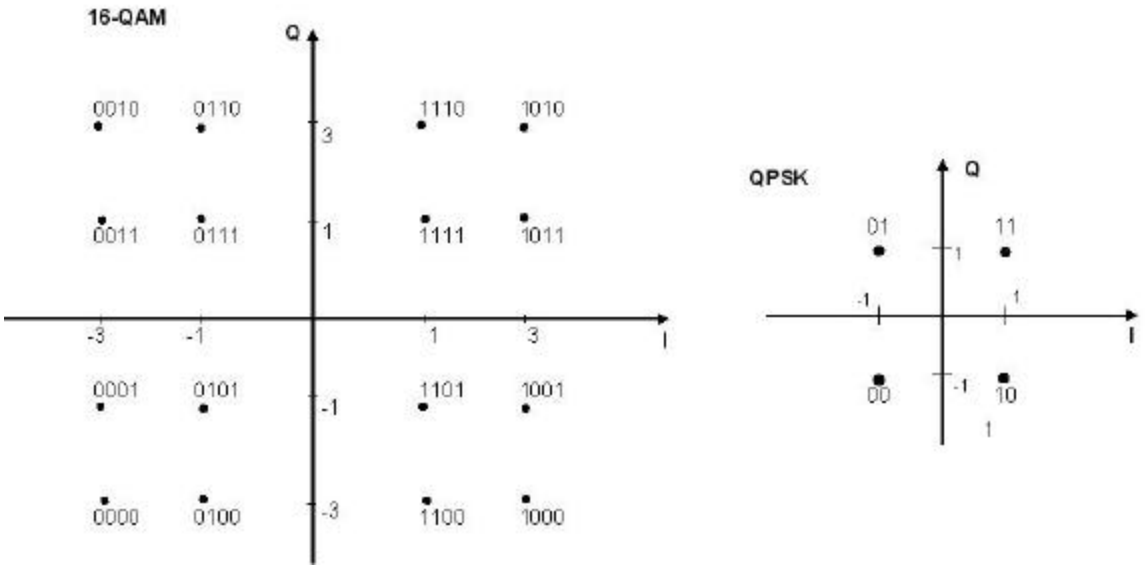


Figure 11. 16-QAM and QPSK Constellations (From Ref. 11.)

d. Differential Quadrature Phase Shift Keying (DQPSK) Modulation

After the symbol mapping process, the data can be frequency or time differential encoded. The Matlab implementation details of differential encoding are explained in Appendix A.

Differential (noncoherent) modulation techniques decode the incoming symbols by comparing the received symbol with the preceding symbol. If the transmitted waveform is given as [17]

$$s(t)=\sqrt{\frac{2E}{T}}\cos\left[w_0 t+q_i(t)\right] \qquad i=1,\ldots,M \qquad 0\le t\le T, \tag{0.1}$$

then the received signal is given by

$$r(t)=\sqrt{\frac{2E}{T}}\cos\left[w_0 t+q_i(t)+a\right]+n(t) \qquad i=1,\ldots,M \quad 0\le t\le T \tag{0.2}$$

where a is assumed to be a random variable uniformly distributed between 0 and $2p$ and $n(t)$ is an additive white Gaussian noise (AWGN) process.

For noncoherent detection, matched filters cannot be used because the matched filter output becomes a function of the unknown phase a. However, assuming that a changes slowly relative to the two symbol period times, the phase difference of two respective signals becomes independent of a and given by [17]

$$\left[q_k(T_2)+a\right]-\left[q_j(T_1)+a\right]=q_k(T_2)-q_j(T_1)=f_i(T_2), \tag{0.3}$$

which suggests that the phase of the previous signaling interval can be used as a phase reference to demodulate the current signal. Of course, this requires the use of differential encoding in the transmitter.

Even though DPSK performs less efficiently than PSK, it was chosen due to the ease of implementation.

e. IFFT

The interleaved and differentially encoded complex frequency array is next modulated onto a desired number of sub-carriers (52 sub-carriers are used in the IEEE 802.11a standard) by implementing an IFFT.

The OFDM baseband sub-carrier is

$$f_k(t) = e^{j2p f_k t} \tag{0.4}$$

where f_k is the k^{th} sub-carrier frequency. An OFDM symbol consists of N modulated sub-carriers. The OFDM signal not including a cyclic prefix is given by [3]

$$s(t) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} z_k f_k(t) \qquad 0 < t < NT \tag{0.5}$$

where z_k is the k^{th} complex data symbol and NT is the OFDM symbol duration. The sub-carriers in Equations 2.4 and 2.5 have frequencies

$$f_k = \frac{k}{NT} \tag{0.6}$$

in the sense that

$$\frac{1}{NT} \int_0^{NT} f_k(t) f_m^*(t)\, dt = d_{k,m} \tag{0.7}$$

ensures orthogonality. If the signal $s(t)$ is sampled with a sampling period of T, the following is obtained:

$$s(n) = s(t = nT) = \text{IDFT}\{z_k\} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} z_k e^{j2p kn/N} \qquad n = 0,1 \ldots N-1. \tag{0.8}$$

This last equation is $\text{IDFT}\{z_k\}$ and was proposed by [18]. As can be seen from Equation 2.8, a baseband OFDM transmission symbol is an N-point complex modulation sequence. It is composed of N complex sinusoids, which are modulated with $z(k)$.

f. Guard Interval Addition

The addition of the Guard interval, T_{GI}, is the next step in generating OFDM symbols. The guard interval is added to the start of OFDM symbols at the transmitter to eliminate ISI and ICI, and it is deleted before the FFT process at the receiver.

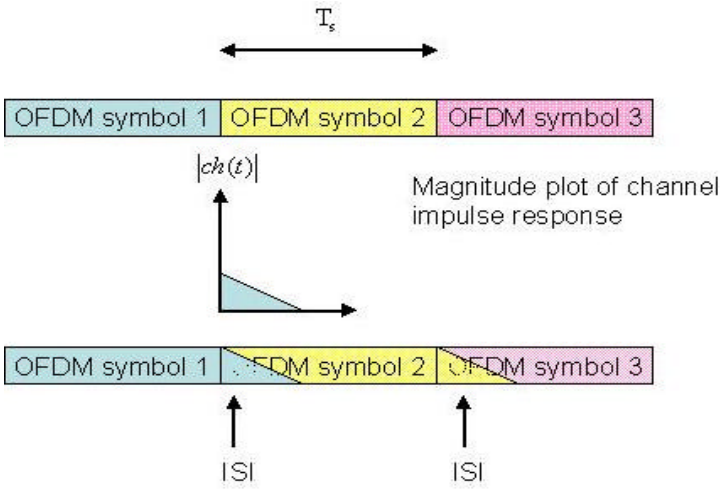
The interval T_{GI} is an important parameter in the IEEE 802.11a standard because it determines the choice of other parameters. It is normally desired that T_{GI} be larger than the expected multi-path delay spread to combat ISI. However, the larger T_{GI},

the smaller the effective symbol duration, thereby resulting in reduced throughput. In the IEEE 802.11a standard, the total OFDM symbol duration is chosen to be $4\ \mu s$, and T_{GI} is $0.8\ \mu s$. Therefore, the effective symbol duration is $3.2\ \mu s$, and the inverse of the effective symbol duration provides the sub-carrier spacing of 0.3125 MHz. These time related parameters of OFDM are listed in Table 1.

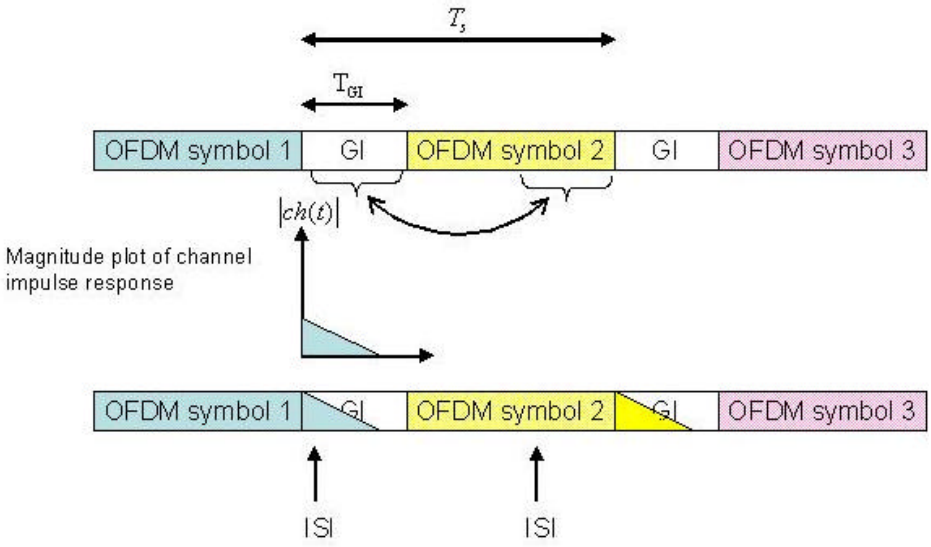
The expression for a low-pass OFDM signal is given by Equation 2.8. The transmitted time domain signal is then obtained by including the guard interval. The analog signal after the digital to analog converter (DAC) in the block diagram is expressed as

$$\hat{s}_{lp}(t) = \frac{1}{N}\sum_{k=0}^{N-1} z(k)\cdot e^{j2pk\Delta f\,(t-T_{GI})} \qquad (0.9)$$

where Δf is sub-carrier spacing. The guard interval T_{GI} is created by appending the IFFT sequence with a circular extension (called cyclic prefix) in order to maintain orthogonality. For example, the cyclic prefix for the transmission sequence $\{s_{lp}(n)\}_{n=0}^{N-1}$ consists of the last m samples $\{s_{lp}(N-m), s_{lp}(N-m-1),\ldots s_{lp}(N-1)\}$, which are added to the beginning of the sequence $\{s_{lp}(0), s_{lp}(1),\ldots s_{lp}(N-1)\}$. The new sequence now consists of $N+m$ samples. The benefit of the guard interval can be observed in Figure 12. Figure 12b illustrates that the decaying transient of symbol 1 in the guard interval can be allowed since it does not affect symbol 2.



(a) OFDM Symbols without Guard Interval



(b) OFDM Symbols with Guard Interval

Figure 12. Effects of Guard Interval (After Ref. 16.)

The disadvantage of the guard interval is that it causes a loss of bandwidth efficiency since data are not transmitted during the guard interval. On the other hand, by adding the guard interval, ICI can be avoided since orthogonality is ensured, and ISI is avoided since adjacent symbols do not interfere with each other.

g. Number of Sub-carriers

The number of sub-carriers is another important parameter in the design of OFDM systems along with the system bandwidth $B \approx 1/T$, the sub-carrier bandwidth $1/NT$, and the guard interval T_{GI}.

Initially, the guard interval T_{GI} is chosen to be larger than delay spread t, and the OFDM symbol length should be larger than T_{GI}. Therefore $NT \gg t$, where N is the number of sub-carriers which yields $N \gg Bt$. However, a large N means smaller frequency spacing between OFDM carriers, which in turn, makes the system vulnerable to Doppler spread and ICI. This last restriction forces larger frequency spacing than the Doppler spread $(B/N \gg f_d)$. As a result, the number of sub-carriers can be chosen in the interval given by [3]

$$t\,B \ll N \ll \frac{B}{f_d}. \tag{0.10}$$

The number of sub-carriers is fixed at 52 in the IEEE 802.11a standard. Four of these sub-carriers are used as a reference to minimize frequency and phase shifts of the signal during transmission.

h. Symbol Pulse Shaping (Windowing)

The time dispersion can be combated by the addition of guard interval. However, it is also necessary to combat the slow decay of OFDM symbols in the frequency domain. The effects of slow decay into the adjacent bands can be avoided by narrowing the spectrum.

The method of choice to narrow the spectra is to use pulse shaping of the OFDM symbol. Pulse shaping is accomplished by using either a time window or a filter [3]. Windowing helps to smooth the transition between symbols and narrows the signal spectrum.

i. RF Modulation

After the pulse shaping process, the OFDM symbols are upconverted to a radio frequency carrier, amplified and transmitted through the antenna. The IEEE 802.11a standard specifies the 5-Ghz band for the carrier frequency. In this thesis, the communication system was simulated at the baseband.

4. OFDM Receiver

A block diagram of the OFDM Receiver module is presented in Figure 13.

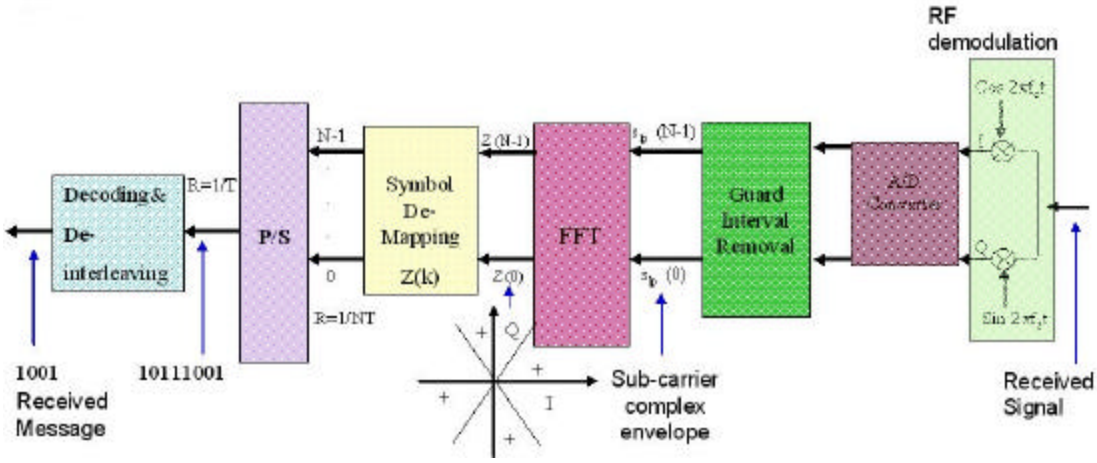


Figure 13. Schematic Diagram of an OFDM Receiver

The processes performed in the transmitter are simply reversed in the receiver. In addition, the receiver must include algorithms for synchronization and channel estimation. In particular the synchronization block performs operations, such as packet detection, timing estimation, frequency offset estimation and correction.

a. Removing Guard Interval and FFT Processing

Since only the baseband processing is being used in this thesis, the down-conversion process in the receiver is not needed. The first step is to remove the guard interval to obtain the information portion of the symbol for further processing. Next, the time domain samples are transformed into the frequency domain by the FFT process. This also makes it possible to recover the OFDM frequency tones.

b. Decoding and Deinterleaving

The next step in the receiver is the time or frequency differential decoding. Following the differential decoding, the inverse mapping of each received complex modulation value into a corresponding N-ary symbol is accomplished. The message is next deinterleaved according to the particular interleaving configured in the transmitter.

c. Received message and Viterbi Decoding

After being exposed to the effects of different channel models, some errors may occur in the transmitted message. The received bit stream is determined after the Viterbi decoder.

The Viterbi decoders are used to decode bit streams encoded by convolutional encoders. After receiving the bit stream from the channel, the Viterbi algorithm searches the trellis to find the path that can create the bit stream closest to the received one. The measure of closeness is different for hard and soft decision decoding. For the hard decision decoding, the Viterbi algorithm tries to find the bit stream path that has the minimum Hamming distance from the received sequence. For soft decision decoding, the algorithm tries to find the bit stream path that has the minimum Euclidian distance from the received sequence. The details of the Viterbi decoding algorithm can be found in [17].

C. SUMMARY

An introduction to OFDM was presented in this chapter. The transmitter and receiver block diagrams have been discussed in detail and the chapter concluded by listing the advantages and disadvantages of OFDM and mentioning the potential problematic areas in OFDM implementation.

The advantages of OFDM can be summarized as follows. OFDM changes frequency selective fading to flat fading and offers high bit rate transmission. Its flexible architecture allows the use of combinations of various coding schemes (rates) and modulation schemes (signal constellations) according to channel condition. OFDM eliminates ISI, is bandwidth efficient, and supports several multiple access schemes (TDMA, FDMA, MC-CDMA, etc.) [15].

On the other hand OFDM is very sensitive to frequency errors, has a large peak to average ratio (PAR), and is sensitive to channel fading [15]. The effect of PAR is analyzed in [19].

OFDM has some potential problematic areas that should be considered during implementation. The first is synchronization, including the time and frequency synchronization. The second is the non-constant power envelope, which requires the use of linear amplifiers. The last problematic area is the requirement of channel estimation since the channel is time-variant [13].

The next chapter presents the steps to create an OFDM signal. OFDM systems' unique requirements, such as IFFT/FFT algorithms and the cyclic prefix, are also covered. Following that, the effects of frequency errors are analyzed.

THIS PAGE INTENTIONALLY LEFT BLANK

III. FREQUENCY ERRORS IN OFDM AND THEIR EFFECTS

Before an OFDM receiver demodulates the sub-carriers, it performs two synchronization tasks. First, it determines the symbol boundaries and the optimal timing instants in order to minimize both the ICI and the ISI. Second, it tries to estimate and correct the frequency errors [20]. OFDM's sensitivity to these frequency errors is one of its main drawbacks. This chapter discusses the effects of these frequency errors on OFDM systems.

A. FREQUENCY ERRORS

The orthogonality of the sub-carriers can be ensured only if the receiver and the transmitter have the same reference frequency. Any deviation from this reference frequency may cause ICI and loss of orthogonality. Another frequency error factor is phase noise, which is caused by random jitter of the phase of the steady sinusoidal waveform generated by the oscillators [20].

Typically frequency errors are generated by the fact that the oscillators in the modulator and demodulator do not have exactly the same frequency.

For single-carrier systems, the effect of phase noise and frequency offset appear only as degradation in the received SNR, rather than ISI or ICI. Nevertheless, many efficient techniques to minimize the effects of this drawback have been proposed in the literature [20, 21].

Other reasons for frequency errors include Doppler shift caused by the relative movement between the receiver and the transmitter, and phase noise introduced by non-linear channels.

Figure 15 shows the front end of an OFDM receiver where most of the frequency errors occur, i.e., the local oscillators and the sample clock at the analog to digital (A/D) converter [3]. The A/D converter causes errors when the receiver does not have the same sample clock frequency as at the transmitter.

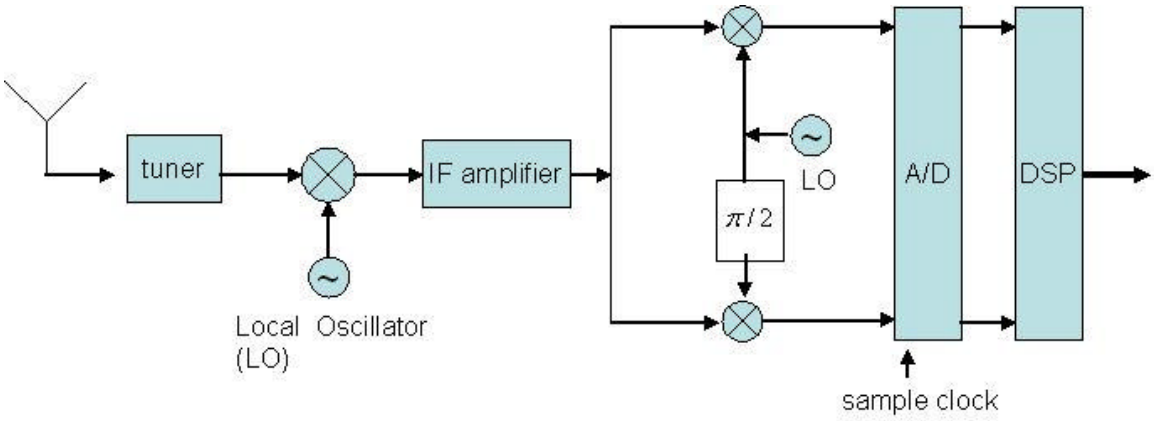


Figure 14. OFDM Receiver Front End (After Ref. 3.)

The frequency offset and the phase noise cause a phase rotation of the received symbols. Coherent OFDM systems need a phase tracking device to obtain the phase of the incoming symbols for correct demodulation [22]. Typically, three types of algorithms are used to estimate the frequency offsets, i.e., to track the phase, in coherent OFDM systems:

- data-aided algorithms that use special training information (pilot tones) within the transmitted signal,
- non-data-aided algorithms that analyze the received signal in the frequency domain, and
- algorithms that use the cyclic prefix of the OFDM signals [21].

For wireless applications, the first method is the method of choice. The implementation details of these coherent OFDM detection algorithms can be found in [21]. It should be noted that in many applications as well as in this thesis, the differential detection technique is used since it does not require channel estimation and the use of pilot tones, even though it causes a loss of SNR.

In the following sections, the effects of each of the aforementioned reasons that cause frequency errors are analyzed. A simple representation of an OFDM signal is created for the ease of analysis.

B. CREATING A SIMPLE REPRESENTATION OF AN OFDM SIGNAL TO SIMPLIFY ANALYSIS

A baseband OFDM signal can be represented by [7]

$$b(t) = \sum_{i=1}^{N-1} A_i \cos(w_i t + f_i) \tag{0.11}$$

where A_i is the amplitude, $w_i = 2p\ f_i$ is the angular frequency, f_i is the phase of the i^{th} sub-carrier, and N is the number of sub-carriers.

According to the modulation technique to be used, either A or f is determined by the data. As discussed in Chapter II, the baseband OFDM signal $b(t)$ is modulated next, onto a RF carrier with frequency f_c:

$$\begin{aligned} s(t) &= 2b(t)\cos w_c t \\ &= 2\sum_{i=0}^{N-1} A_i \cos(w_i t + f_i)\cos w_c t \\ &= \sum_{i=0}^{N-1} A_i \left\{\cos\left[(w_c + w_i)t + f_i\right] + \cos\left[(w_c - w_i)t - f_i\right]\right\} \end{aligned} \tag{0.12}$$

where $w_c = 2p\ f_c$, and we assume the phase of the carrier to be zero for simplicity. Since a single side band transmission is enough to carry the information in A_i or f_i, it is assumed that the upper sideband is used, and therefore the transmitted signal can be represented as [7]

$$s(t) = \sum_{i=0}^{N-1} A_i \cos\left[(w_c + w_i)t + f_i\right]. \tag{0.13}$$

C. THE EFFECT OF DOPPLER FREQUENCY SHIFT, PHASE NOISE AND OSCILLATOR FREQUENCY OFFSET ON THE OFDM SYSTEMS

In this section the theoretical analysis of the effects of frequency errors is presented. The maximum Doppler shift occurs when the two mobile nodes move toward each other, given by [23]

$$f_d = \frac{v f_c}{c} \tag{0.14}$$

where v is the relative speed of the two nodes, f_c is the carrier frequency and c is the speed of light $\left(3\times10^8\,\mathrm{m/s}\right)$.

1. Effect of Doppler Shift on the Carrier Frequencies, Sub-carriers, Envelope and Symbol Timing

An OFDM signal consists of numerous sub-carriers with different frequencies. The amount of Doppler shift affecting the i^{th} sub-carrier is given by [7]

$$(f_c \pm f_i) \xrightarrow{Doppler\ \ shift} (1+x)(f_c \pm f_i) \tag{0.15}$$

where x is the percentage of the change in frequency and is determined by

$$x = \frac{f_d}{f} = \frac{v}{c}\cos q. \tag{0.16}$$

The right-hand side of Equation 3.5 can be written as

$$(1+x)(f_c \pm f_i) = (1+x)\, f_c \pm (1+x)\, f_i \tag{0.17}$$

which demonstrates that the Doppler frequency shift affects the carrier frequency and the sub-carrier frequencies by the same percentage x [7]. The Doppler shift of the carrier frequency can be calculated as

$$f_{dc} = \frac{v f_c}{c}\cos q \tag{0.18}$$

and the Doppler shift of the sub-carrier frequencies as

$$f_{di} = \frac{v f_i}{c}\cos q. \tag{0.19}$$

By using Equation 3.3 again, the transmitted OFDM signal with Doppler shift can be written as

$$\begin{aligned} s(t) &= \sum_{i=0}^{N-1} A_i \cos\left[(1+x)(w_c + w_i)t + f_i\right] \\ &= \sum_{i=0}^{N-1}\left\{A_i \cos\left[(1+x)w_i t + f_i\right]\cos\left[(1+x)w_c t\right] - A_i \sin\left[(1+x)w_i t + f_i\right]\sin\left[(1+x)w_c t\right]\right\}. \end{aligned} \tag{0.20}$$

In Equation 3.10, $A_i \cos\left[(1+x)w_i t + f_i\right]$ can be thought of as the envelope of the carrier, $\cos\left[(1+x)w_c t\right]$, which helps to demonstrate that the Doppler shift affects the envelope and the carrier frequency by the same percentage [7]. The Doppler shift also affects the symbol rate and the time synchronization.

2. Phase Noise, Oscillator Frequency Offset and Their Effects

The carrier frequency offset has two destructive effects on OFDM systems. The first is the reduction of the amplitude of the desired sub-carrier and the second is the introduction of ICI [21]. The reduction of the amplitude happens because the desired sub-carrier is not sampled at the peak of the sinc function of the DFT since the sinc functions are shifted (see Figure 15). Also, the ICI is caused since orthogonality is lost between the neighboring sub-carriers [24].

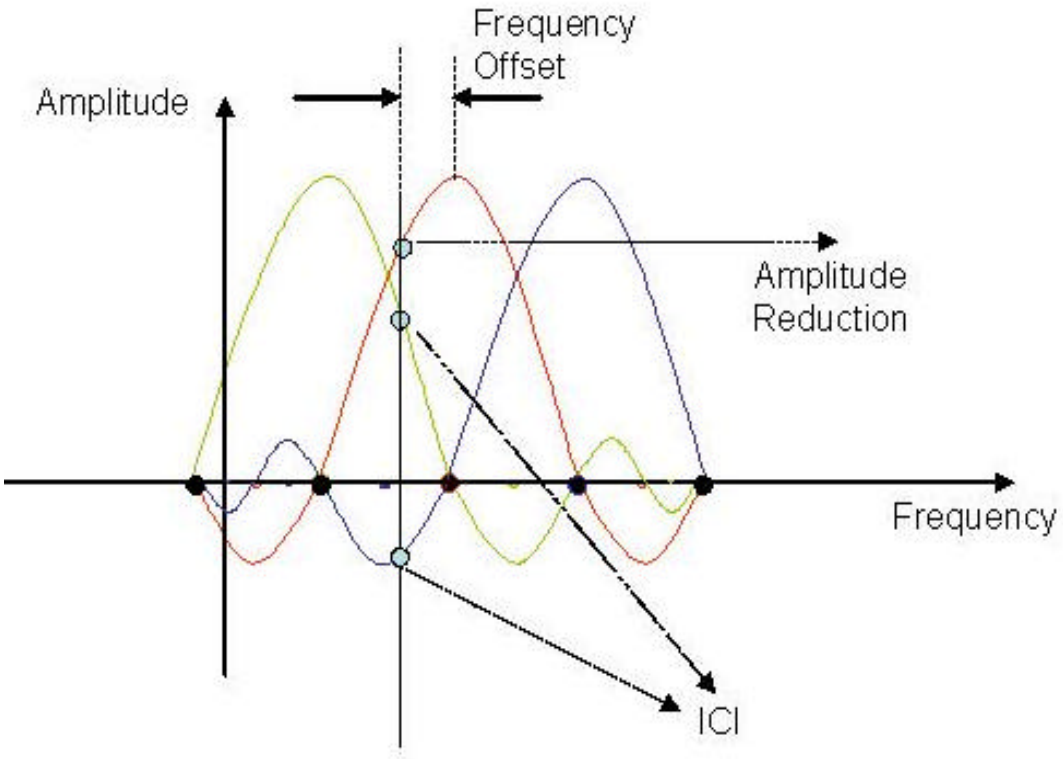


Figure 15. Effects of Frequency Offset: Reduced Amplitude and ICI (After Ref. 24.)

Moose [5] and Pollet *et al*. [4] analytically evaluated the effects of the carrier frequency offset and carrier phase noise on the SNR degradation for an AWGN channel.

The results derived in these papers are used for many of the studies on frequency errors. As a result, we have used these results as benchmarks for the simulation results in this thesis.

Moose [5] models the channel by a complex transfer function H_k for the k^{th} sub-carrier and assumes that a frequency offset of e exists for all sub-carriers. The frequency offset, e, is usually expressed relative to the data symbol rate, and it includes both the Doppler shift and the local oscillator frequency offset effects. As mentioned previously, each sub-carrier is affected by the Doppler shift according to its frequency value. Taking all the causes into consideration, the total offset for each sub-carrier after down conversion can be expressed as $\Delta f_i = f_{dc} + f_{di} + \Delta f_{LO}$ where f_{dc} is the carrier Doppler shift, f_{di} is the i^{th} sub-carrier Doppler shift and Δf_{LO} is the local oscillator frequency offset [7].

As can be seen from Equations 3.8 and 3.9, $f_{dc} \gg f_{di}$ since $f_c \gg f_i$. Thus, $\Delta f_i \cong f_{dc} + \Delta f_{LO} = \Delta f$. This result demonstrates that the frequency offset is independent of the sub-carriers, which in turn states that the relative offset $e = \Delta f / R_s$ is also independent of sub-carriers [7].

The SNR calculated at the output of the DFT of the receiver is given by [5]

$$\text{SNR} \geq \frac{E_c}{N_o}\left(\frac{\sin pe}{pe}\right)^2 \frac{1}{1+0.5947\left(\frac{E_c}{N_o}\right)(\sin pe)^2} \quad ,|e|\langle 0.5 \tag{0.21}$$

where E_c/N_o is the OFDM carrier energy to the AWGN spectral density. An upper bound on the degradation can be obtained from Equation 3.11 as follows:

$$D_{\text{dB}} \leq 10\log_{10}\left(\frac{1+0.5947\frac{E_s}{N_o}\sin^2 p\Delta f}{\sin c^2 \Delta f}\right) \tag{0.22}$$

where the factor 0.5947 is derived by taking the lower bound of the summation of all interfering sub-carriers [24].

Similar results to Equations 3.11 and 3.12 are found in [4] in which the channel is modeled by a time-varying phase $q(t)$, which is a result of either the phase noise of the carriers or a carrier offset between the receiver and transmitter. A distinction is made between phase noise and frequency offset.

a. Phase Noise

For the phase noise case, $q(t)$ is assumed to be a Wiener process with $E\{q(t)\} = 0$ and $E\left\{\left(q(t_o + t) - q(t_o)\right)^2\right\} = 4\pi b|t|$, where b (Hz) is the one-sided 3-dB linewidth of the Lorentzian power density spectrum of the free-running carrier generator [4].

Phase noise has two main effects. First, it causes a random phase variation common to all sub-carriers. The effects of this common phase error are minimized by employing phase tracking techniques or differential decoding. Second, it introduces ICI. Based on the model defined in [4], the degradation D in SNR, i.e., the required increase in SNR to compensate for the phase noise is

$$D_{\text{dB}} \cong \frac{11}{6 \cdot \ln 10}\left(4\pi N \frac{b}{R}\right)\frac{E_s}{N_o}. \tag{0.23}$$

Since $R = N/T = NR_s$, where N is the total number of sub-carriers and R_s is the sub-carrier symbol rate, Equation 3.13 can be rewritten as

$$D_{\text{dB}} \cong \frac{11}{6 \cdot \ln 10}\left(4\pi \frac{b}{R_s}\right)\frac{E_s}{N_o}. \tag{0.24}$$

The plot of SNR degradation in dB as a function of the normalized bandwidth (b/R_s) is shown in Figure 18 for QPSK. This figure is plotted for $E_s/N_o = 10.5$ dB corresponding to a BER of 10^{-6} for uncoded QPSK [20].

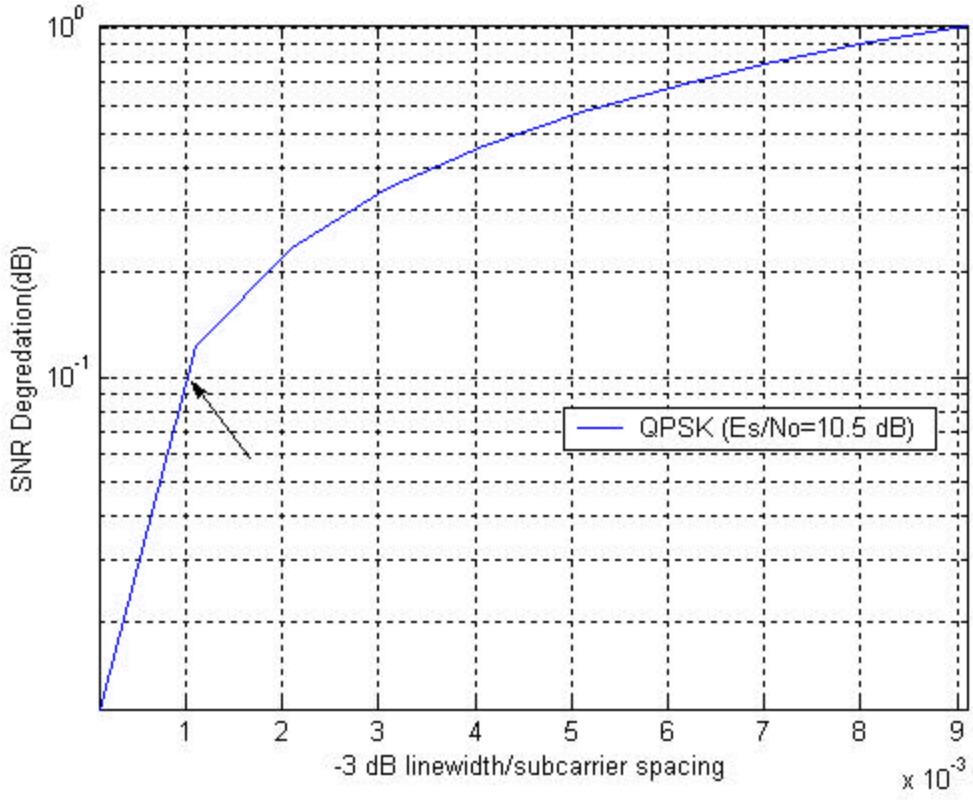


Figure 16. SNR Degradation Versus - 3-dB Bandwidth of the Phase Noise Spectrum for QPSK.

By observing Figure 16, it can be concluded that for a negligible SNR degradation of about 0.1 dB, the - 3-dB phase noise bandwidth should be approximately 0.1 percent of the sub-carrier spacing for QPSK. However, this value changes depending on the modulation.

b. Frequency Offset

For the frequency offset case, $q(t)$ is given by

$$q(t) = 2p\Delta F\, t + q_o \tag{0.25}$$

where ΔF is the carrier offset. If there is a frequency offset, then the number of cycles in the FFT interval is not an integer and this causes ICI after the FFT. The sub-carriers in the middle of the OFDM spectrum are more exposed to ICI than the sub-carriers on the edges of the spectrum because they have interfering sub-carriers on their both sides [20].

Based on the model developed in [4], the degradation D in SNR for frequency offset is given by

$$D_{\text{dB}} \cong \frac{10}{3 \cdot \ln 10} \left(p N \frac{\Delta F}{R} \right)^2 \frac{E_s}{N_o}. \tag{0.26}$$

By using the relation $R = N/T = NR_s$, Equation 3.16 can be rewritten as

$$D_{\text{dB}} \cong \frac{10}{3 \cdot \ln 10} \left(p \frac{\Delta F}{R_s} \right)^2 \frac{E_s}{N_o} \tag{0.27}$$

where $\Delta F / R_s$ is the relative frequency offset. The plot of SNR degradation in dB as a function of the relative frequency offset $(\Delta F / R_s)$ is shown in Figure 17 for QPSK. This figure is plotted for $E_s / N_o = 10.5$ dB corresponding to a BER of 10^{-6} for uncoded QPSK [20].

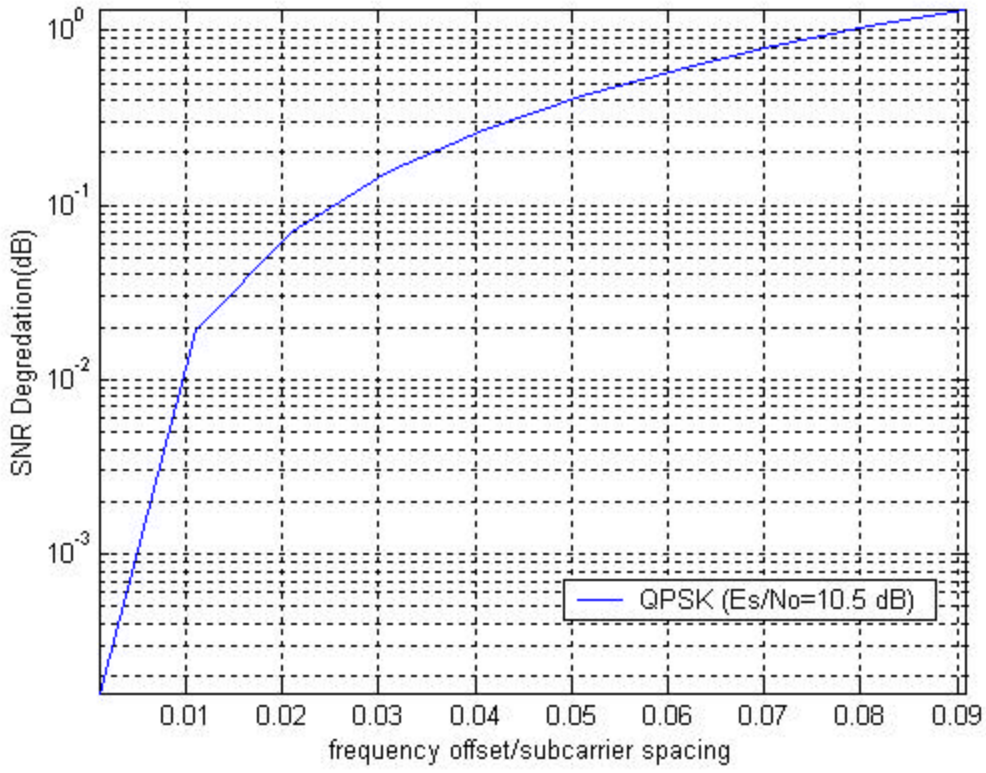


Figure 17. SNR Degradation Versus the Normalized Frequency Offset for QPSK.

From Figure 17, it can be seen that for a negligible SNR degradation of about 0.1 dB, the tolerable frequency offset should be approximately 2.5% of the sub-carrier spacing for QPSK. However, this value changes depending on the modulation.

D. SUMMARY

This chapter discussed the frequency errors and their causes. It was demonstrated that SNR degradation due to the Doppler shift, most of the time, is negligible, and this left frequency offset and phase noise of local oscillators as the main sources of degradation. It was also shown that this degradation varies strongly according to the modulation used. In [21], it was illustrated that 64-QAM could not tolerate more than 1% carrier frequency error for a SNR degradation of 0.5 dB, whereas QPSK modulation could tolerate 5% carrier frequency error for the same SNR degradation. It can also be stated that close spacing of carriers in frequency made the tolerable frequency offset a small percentage of the channel bandwidth. In [5], it is illustrated by simulation that the carrier frequency offset is limited to 4% or less of the intercarrier spacing to maintain signal-to-interference ratios of 20 dB or greater for the OFDM carriers.

In the next chapter, this thesis will attempt to simulate an OFDM system under frequency offset and phase noise conditions to observe the expected performance degradation discussed in this chapter.

IV. OFDM SYSTEM SIMULATIONS IN MATLAB

The previous chapters covered the principles of OFDM and discussed the frequency errors that might occur in a communication system as well as the effects of these frequency errors. This chapter presents simulation of an OFDM communication system, operating under different channel conditions. The results of simulation are compared to the theoretical results discussed in Chapter III. The simulation steps performed are presented in Figure 18.

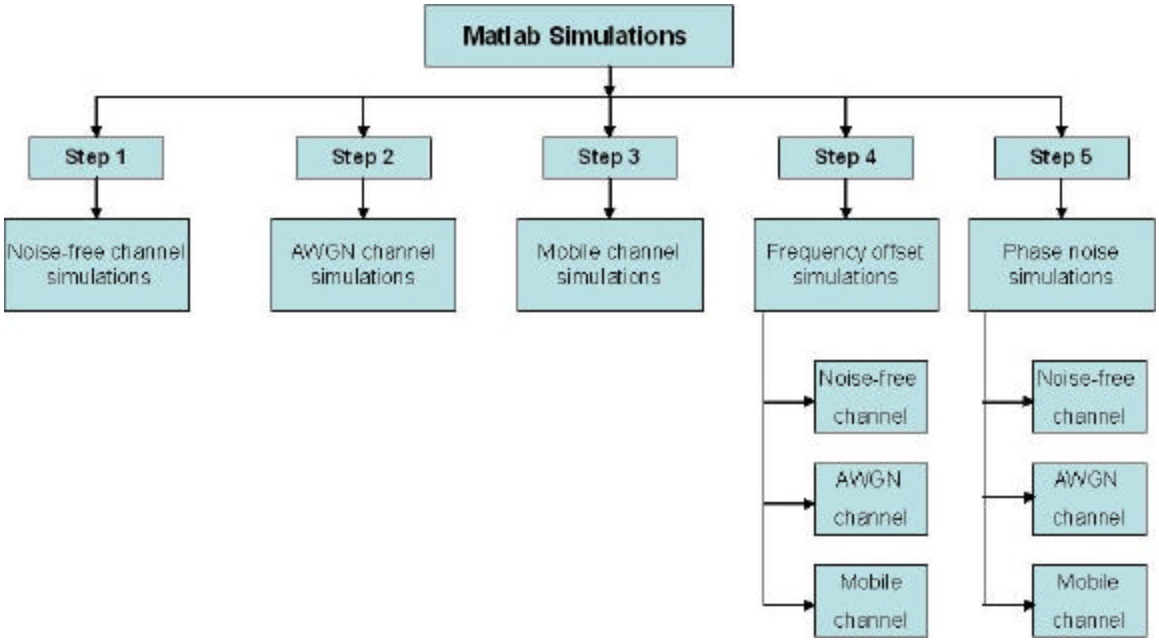


Figure 18. Matlab Simulations Scheme

A. SELECTING SYSTEM PARAMETERS

The simulations attempt to emulate the bit rate and performance requirements of the IEEE 802.11a standard by using the Matlab software. An operational bit rate of 24 Mbps was chosen and, according to the standard, the occupied bandwidth was taken as 16.6 MHz. The constellation to be used was decided based upon these parameters. First, the number of bits/Hz needed to be transmitted to meet the requirements is calculated as

$$\frac{\text{bit}}{\text{Hz}} = \frac{24\text{Mbps}}{16.6\text{MHz}} = 1.44\text{bit/Hz} \approx 2\text{bit/Hz}. \qquad (0.28)$$

QPSK signaling was chosen for this simulation, and the actual bit rate was $2 \times 16.6 = 33.2\text{Mbps}$. Even though this seems much more than the required bit rate, the addition of the guard interval and FEC reduces the bit rate to the desired level [25].

The required symbol interval in the standard was given as $4\,\mu\text{s}$, and the sub-carrier spacing was given in Table 1 as 0.3125MHz. The number of sub-carriers used in the standard is calculated using this sub-carrier spacing and occupied bandwidth as follows:

$$N = \frac{\text{occupied BW}}{\text{frequencyspacing}} = \frac{16.6\text{MHz}}{0.3125\text{MHz}} = 53. \qquad (0.29)$$

The standard actually uses 52 sub-carriers with 4 of these reserved for pilot signals in coherent detection systems to make the system robust against frequency errors. Since differential encoding and decoding were used in this work, all 52 sub-carriers could be used to carry information symbols.

A 16-point guard interval was used during the simulations to represent an 800-ns period, and a 64-point FFT was used to represent the $3.2\ \mu\text{s}$ information signal length, i.e., a total of 80 time samples representing the standard $4\ \mu\text{s}$ OFDM symbol.

B. SIMULATION METHODOLOGY

After modifying the code from [25, 26] and adding required sub-blocks to implement the OFDM system for this thesis, a number of system simulations were performed using different channel models.

The simulation process was performed in five steps. The first step was to verify if the Matlab code was properly functioning. After verification of the code, a channel with only AWGN was simulated, and the system's performance in this channel was investigated. The third step was to investigate the performance of four mobile channels. During this step, how different mobile channels affect the system was demonstrated and, then, the most severe of them was chosen as the mobile channel. The first three simulation steps were conducted to ascertain the system's response to different channel models without introducing any frequency offset and phase noise. The main goals of the simulations were achieved in the fourth and the fifth steps. The fourth step included the fre-

quency offset, which was studied for three different channels. In the fifth step, the phase noise was introduced and the performance measured for three different channels. Table 4 summarizes all of the simulation steps.

Step No.	**Simulation**	**Channel Description**
1.	Code check	Channel without AWGN and multi-path effects (Ideal channel).
2.	Performance in AWGN	AWGN channel.
3.	Performance in Mobile channel	Multi-path+AWGN channel.
4.	Performance with Frequency offset	a. Ideal channel. b. AWGN channel c. Mobile channel
5.	Performance with Phase noise	a. Ideal channel. b. AWGN channel c. Mobile channel

Table 4. Simulation Steps

This step-by-step simulation methodology, beginning with the individual channel simulations and advancing to the combined channel simulations helped to evaluate each channel model's output individually. This approach made it possible to detect the errors caused by incorrectly designed sub-blocks at an early stage and forced a redesign of the incorrectly functioning blocks.

The implementation details and results of each simulation step are explained below.

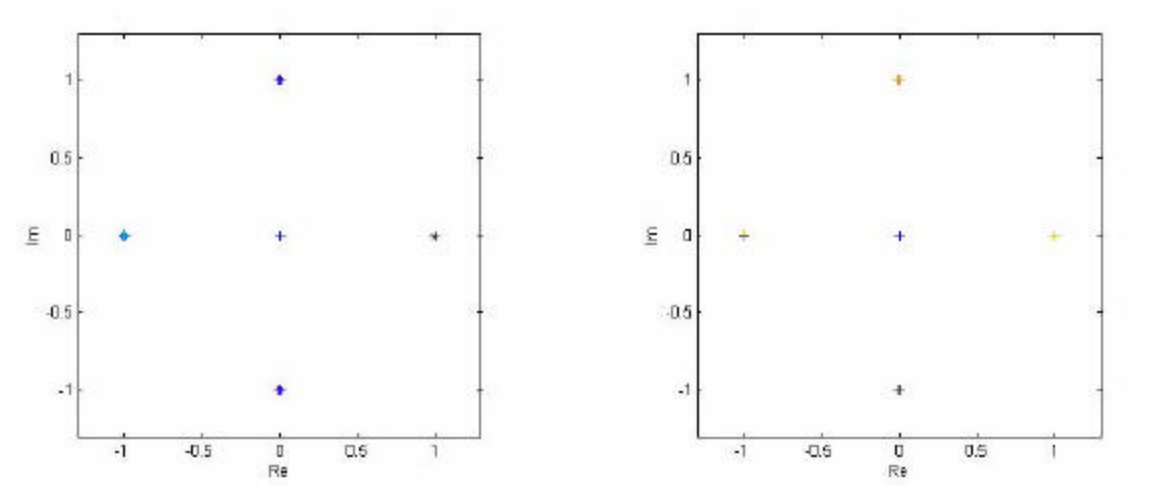
1. Code Check (Channel Model 0)

The first step in the simulation process was to check whether or not the system model functioned satisfactorily in a noise-free, ideal channel. Due to the lack of any channel distortions affecting the signal, the transmitted and received messages should be exactly the same in the case of a correctly implemented overall system. After many simulations, with different parameters each time, it was determined that the system was functioning correctly since it was possible to receive the transmitted signals.

Table 5 lists the parameters of a simulation run for QPSK realization. The transmitted and received QPSK signal constellations under noise-free channel conditions are shown in Figure 19, which verified a correctly functioning system since we were able to receive exactly what we had transmitted.

# of carriers	# of symbols	Channel model	seed	Interleaver matrix
48	300	0	22	[28 12]

Table 5. Code Check Simulation Parameters



(a) (b)

Figure 19. (a) Transmitted and (b) Received QPSK Constellations After Differential Decoding, from a Channel Model 0 Simulation

2. OFDM Performance in an AWGN Channel (Channel Model 1)

In this step, AWGN channel simulations were conducted, and the system's performance results were compared with the reported values in other related studies [25, 26 and 27].

The theoretical symbol error rate P_b for BPSK is given by

$$P_b = Q\left(\sqrt{\frac{E_b}{N_o}}\right) \tag{0.30}$$

where [17]

$$Q(x) = \frac{1}{\sqrt{2p}}\int_x^\infty \exp\left(-\frac{u^2}{2}\right)du.$$

For M-ary modulation, P_b can be approximated by assuming that each symbol error causes only one bit error as follows:

$$P_b \approx \frac{P_s}{\log_2 M} = \frac{P_s}{k} \tag{0.31}$$

where k is the number of bits and $M = 2^k$ is the size of the constellation. For QPSK the symbol error rate is given by [21]

$$P_s = 2Q\left(\sqrt{2\frac{E_b}{N_o}}\right)\left[1-\frac{1}{2}Q\left(\sqrt{2\frac{E_b}{N_o}}\right)\right]. \tag{0.32}$$

For higher order PSK modulation P_s can be approximated by

$$P_s \cong 2Q\left(\sqrt{\frac{E_s}{N_o}}\right)\sin\left(\frac{p}{M}\right) \tag{0.33}$$

where $E_s = E_b\left(\log_2 M\right)$ is the energy per symbol. As seen from Equation 4.4, the bit error rate can always be derived by dividing P_s by the number of bits.

An OFDM system using QPSK and the parameters listed in Table 6 was simulated under AWGN conditions. The received QPSK constellation is shown in Figure 20 for a s value of 0.02.

# of carriers	# of symbols	Channel model	seed	Interlever matrix	s range
48	20000	1	33	[144 139]	0-0.08

Table 6. AWGN Channel Simulation Parameters

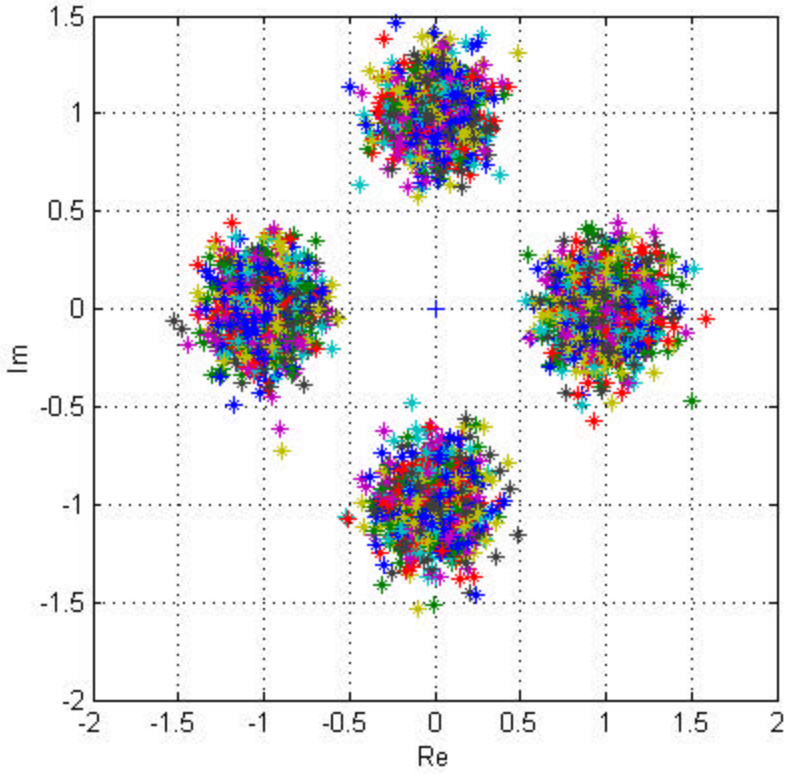


Figure 20. Effect of AWGN on Received Signal Constellation for $s = 0.02$ (Before Differential Decoding)

Figure 21 shows the simulation BER plot under AWGN conditions. Clearly, the errors decrease as the E_b/N_o increased. This result is used in later simulations as a reference for comparing the results with frequency errors (see Figures 29, 30, 34 and 35, discussed later in the thesis).

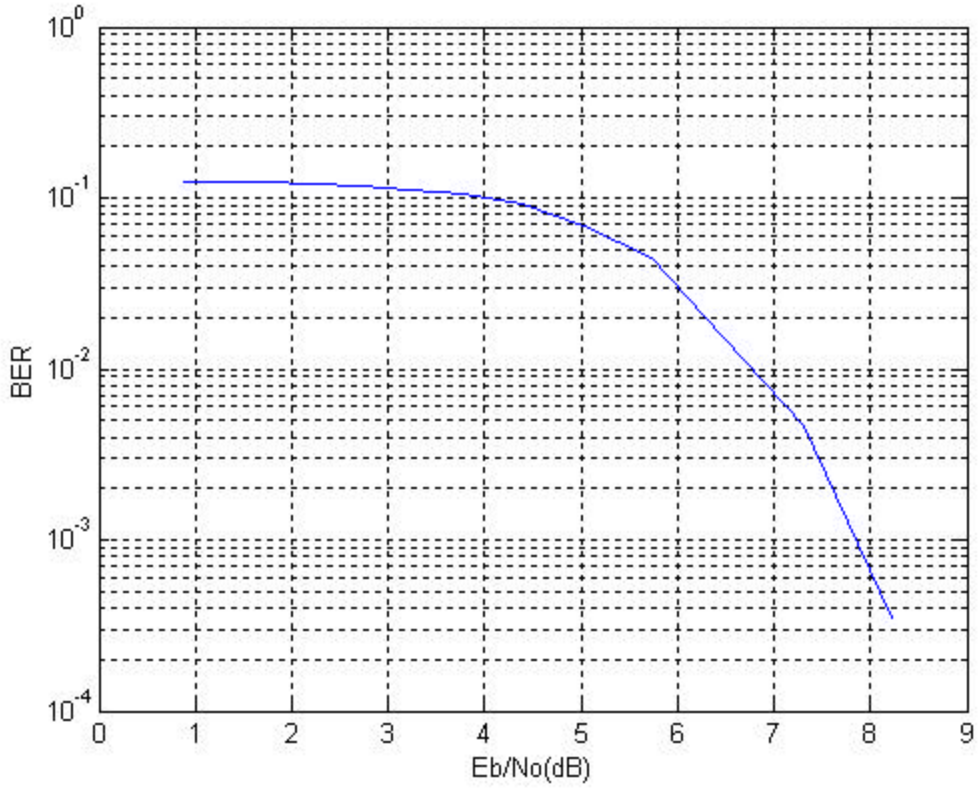


Figure 21. BER versus E_b/N_o Plot for a QPSK Simulation in AWGN Channel

3. OFDM Performance in Mobile Channel (Channel Model 2)

a. Mobile Channels

Mobile channels are composed of both multipath and AWGN components. The multipath component of mobile Channels 1 and 2 has a delay line with 8 taps. The input time domain signal is filtered by 4 parallel 7^{th} order FIR filters, whose coefficients are fixed according to [28]. These filters implemented in m-file *cvdd* as in Figure 22. The FIR filter output is computed using

$$y_i(n) = \sum_{k=0}^{7} a_{ik} x(n-k) \qquad i = 1,2,3,4 \tag{0.34}$$

where a_{ik} are the filter coefficients and $x(n)$ are the time-domain input samples. The filter coefficients used for each of the four FIR filters are listed in Table 7.

k	0	1	2	3	4	5	6	7
a_{1k}	−0.013824	0.054062	−0.157959	0.616394	0.616394	−0.157959	0.054062	0.013824
a_{2k}	0.003143	−0.019287	0.1008	−1.226364	1.226364	−0.1008	0.019287	−0.003143
a_{3k}	0.055298	−0.216248	0.631836	−0.465576	−0.465576	0.631836	−0.216248	0.055298
a_{4k}	−0.012573	0.077148	−0.403198	0.905457	−0.905457	0.403198	−0.077148	0.012573

Table 7. FIR Filter Coefficients for Mobile Channels 1 and 2

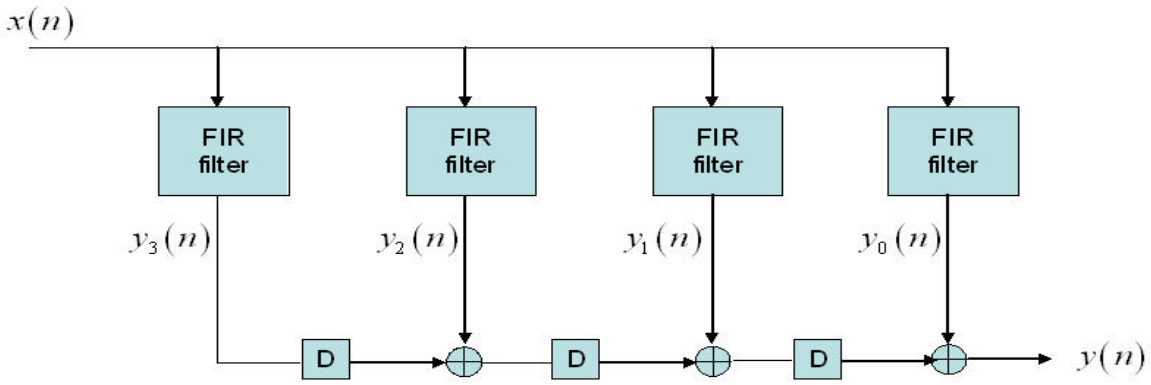


Figure 22. Modeling of Mobile Channels 1 and 2 Using FIR Filters and Delay Elements D.

Multipath components of mobile channels-3 and 4 use 6-tap FIR filters but they have time-varying coefficients with Jakes spectrum [29]. The filtered output is derived simply by summing the delayed paths with the direct path.

All multipath components also include power loss of received signal levels and Doppler frequency shifting characteristics. (The delay, power loss and Doppler vectors can be altered to reflect the desired channel characteristics.)

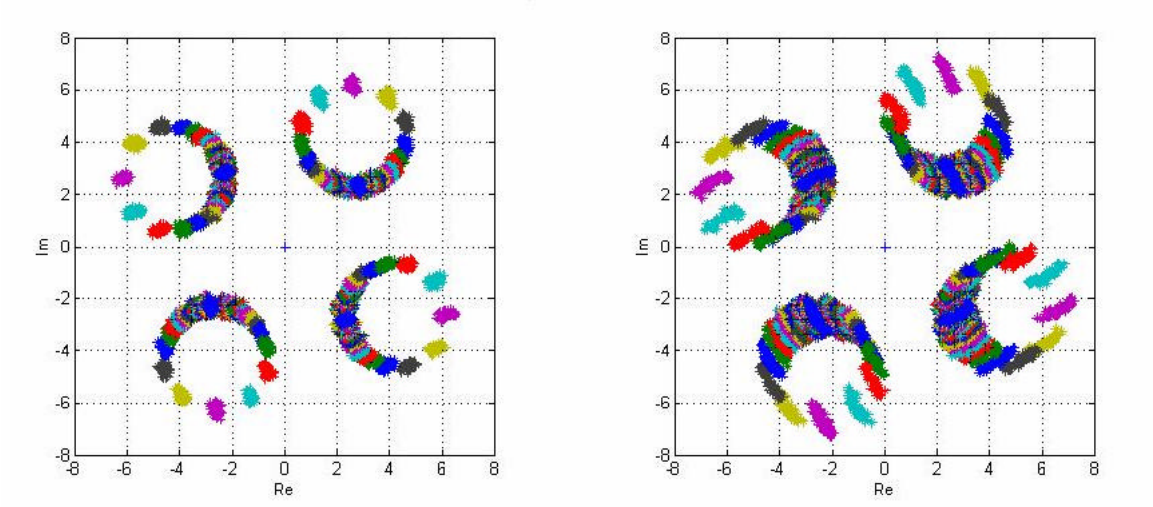
b. Mobile Channel Simulations:

The goal of this simulation step was to see the mobile channel effects on the transmitted symbols. The four different mobile channel models were simulated and the performance curves for each of them presented. The one that causes the most degradation was chosen to be able to simulate the characteristics of a mobile outdoor environment.

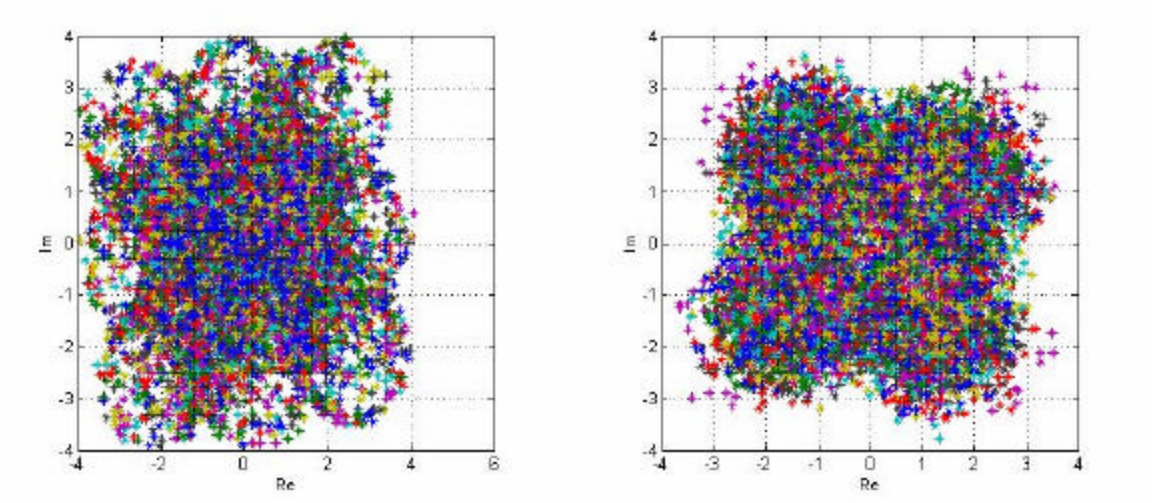
Mobile channel simulations were conducted with the parameters in Table 8. Figure 23 shows the received signal constellations after each mobile channel before differential decoding for a noise variance value of $s = 0.01$. Since the mobile channel causes constructive and destructive effects on the OFDM symbols, the received constellation points are scattered from their transmitted positions. If the differential encoding/decoding were not implemented, a large number of received symbols would be decoded in error since many points are scattered into adjacent phase sectors.

# of carriers	# of symbols	Channel model	seed	Interlever matrix	*s*
48	20000	21,22,23,24	33	[144 139]	0.01

Table 8. Mobile Channel, Simulation 1 Parameters



(a) (b)



(c) (d)

Figure 23. Effect of Mobile Channels on Received Signal Constellations (Before Differential Decoding). (a) Mobile Channel 1 (b) Mobile Channel 2 (c) Mobile Channel 3 (d) Mobile Channel 4.

Figure 24 shows the received QPSK constellations after differential decoding. The scattered points are placed within their previous phase sectors, thereby reducing the number of errors. Figure 25 illustrates the effect of the mobile channels on the magnitude plot of the received symbols. As previously discussed, the sub-carriers in the middle of the OFDM spectrum experience more multipath loss than the sub-carriers on the edges of the spectrum because the sub-carriers in the middle have interfering sub-carriers on both sides. The effects of multipath can be observed in the form of magnitude distortion in Figure 25. Also, we can see the randomly generated peaks, which can be attributed to the use of the Jakes spectrum for mobile Channels 3 and 4.

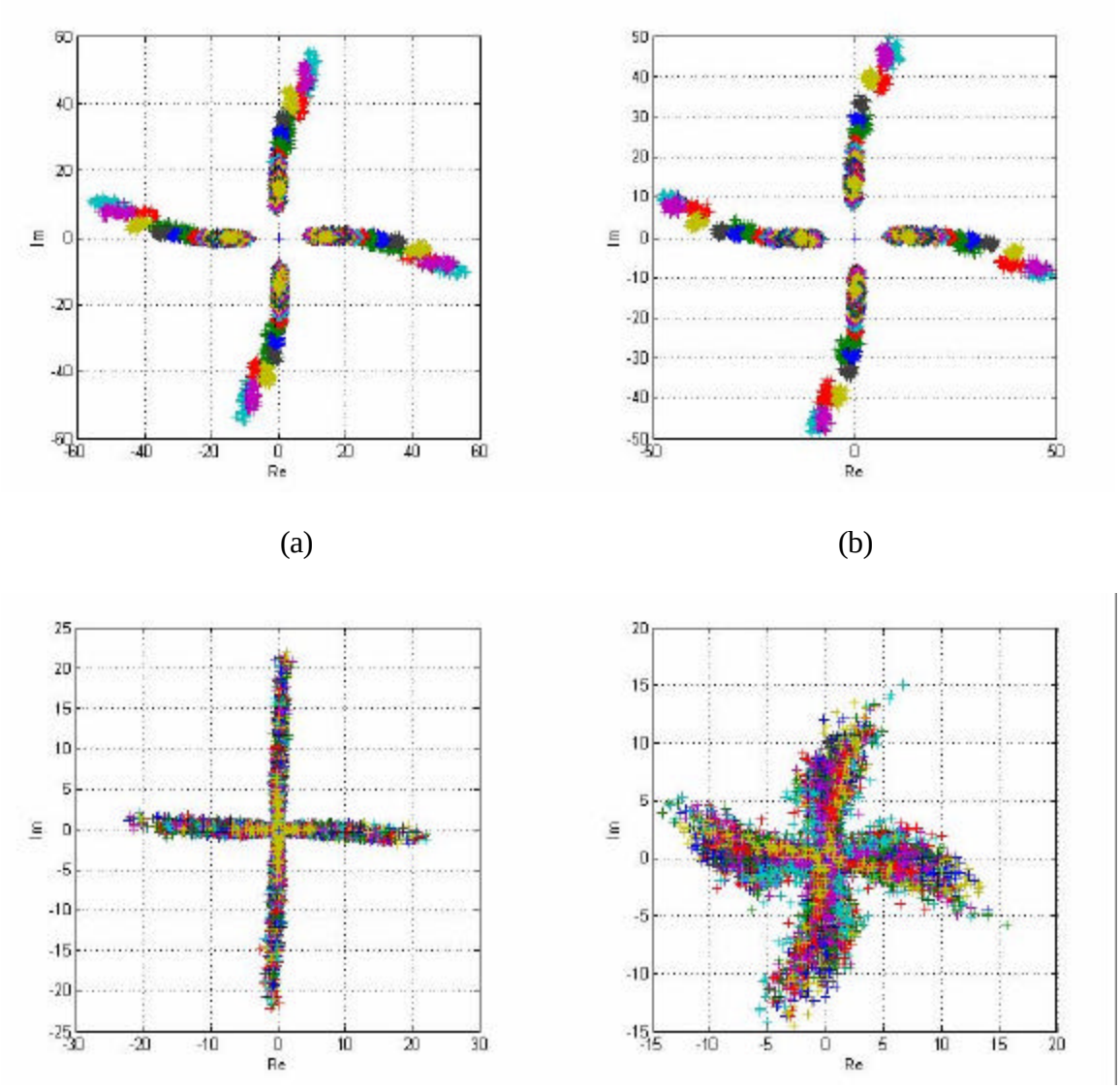


Figure 24. Received Signal Constellation Affected by Mobile Channels (After Differential Decoding). (a) Mobile Channel 1 (b) Mobile Channel 2 (c) Mobile Channel 3 (d) Mobile Channel 4.

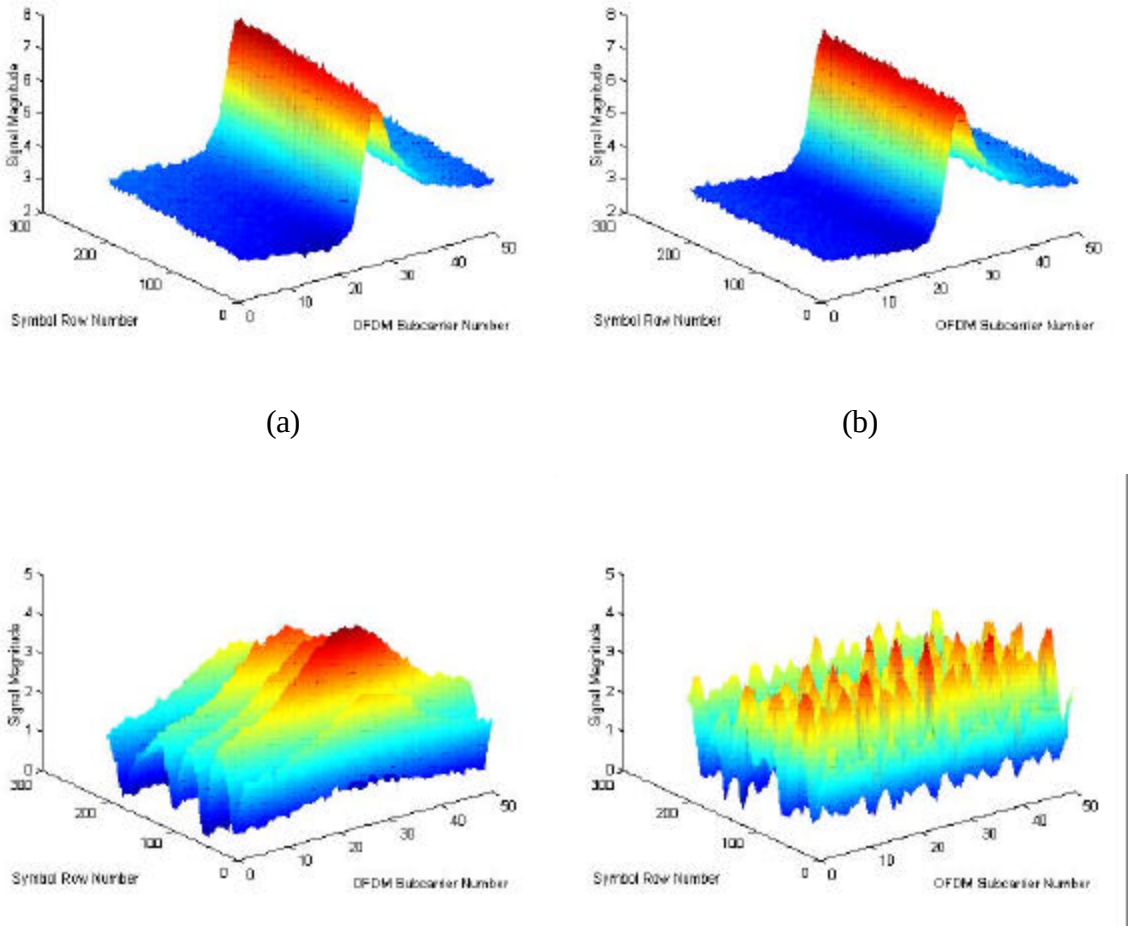


Figure 25. Effect of Mobile Channels on the Magnitude Plot of Received Symbols. (a) Mobile Channel 1 (b) Mobile Channel 2 (c) Mobile Channel 3 (d) Mobile Channel 4.

BER plots of the OFDM system under the mobile channel conditions were obtained by configuring the system with the parameters in Table 7. A range of s values from 0 to 0.08 was used to obtain these plots.

Figure 26 shows the BER versus E_b/N_o plot of each mobile Channel. By observing the BER curves, it can be seen that mobile Channel 1 and 2 required more signal energy relative to the other channel models for the same BER. Mobile Channels 1 and 2 had approximately the same performance, and we randomly chose mobile Channel 1 as

the mobile channel model. This figure justifies that Doppler frequency shift have negligible effects on the performance of OFDM systems since we get similar BER curves for mobile Channel 1 and 2. The Doppler frequency for mobile Channel 1 was 15 Hz and for mobile Channel 2 it was 5 Hz, according to [29].

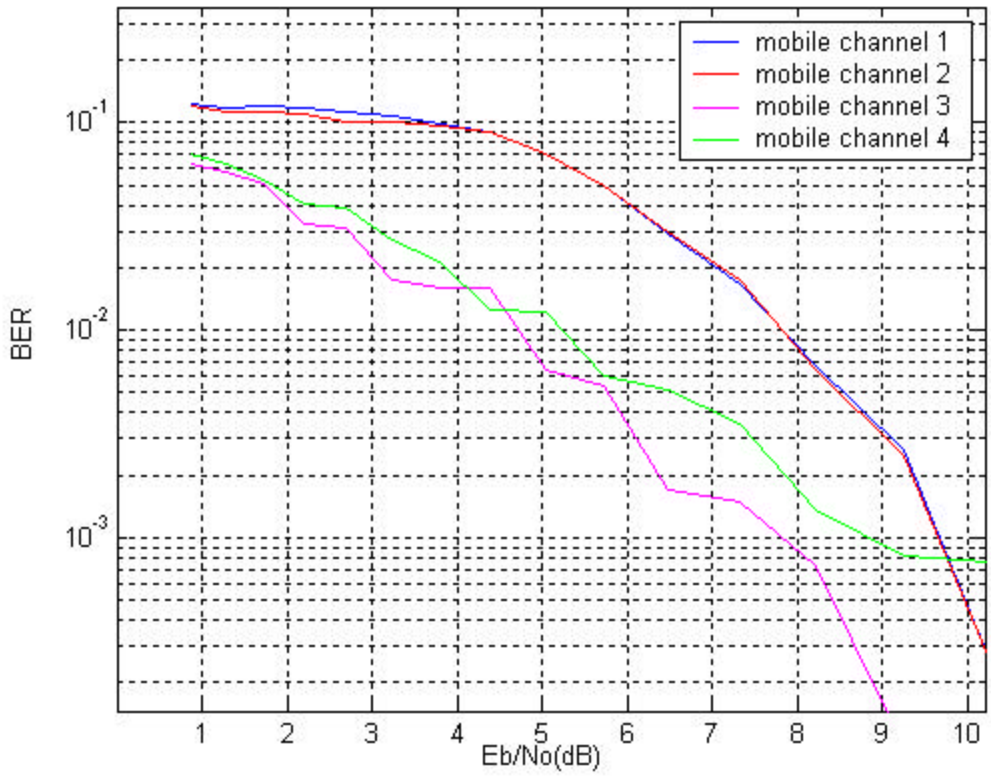


Figure 26. BER versus E_b/N_o Plots for QPSK Simulation in Mobile Channels.

4. Performance with Frequency offset

a. Noise Free Channel

This step of the simulation process was to verify if it were possible to implement the frequency offset. As previously mentioned in Chapter III, for the frequency offset case, $q(t)$ is deterministic and is calculated from $2p\,\Delta F\,t + q_o$, where ΔF is the carrier offset. In the literature, however, frequency offset is usually expressed as a ratio of the frequency spacing $(\Delta F/R_s)$. The performance degradation caused by frequency offset

is also expressed in terms of this relative frequency offset. Therefore, in the simulations, the same approach was used, and each time domain sample of the OFDM symbols was exposed to the effective value of the relative frequency offset.

To see the frequency offset effects exclusively, a noise-free channel was needed. Thus, Channel 0 was chosen. The easiest way to observe the effect of frequency offset was to compare the signal constellations of the transmitted and received symbols. The code to simulate a channel with only frequency offset was configured with the parameters in Table 9. A relatively small frequency offset value was chosen in order to observe its effect clearly.

# of carriers	**# of symbols**	**Channel model**	**seed**	**Interleaver matrix**	$\Delta F / R_s$ **(%)**
48	8000	0	22	[167 48]	0.1

Table 9. Frequency Offset in Noise Free Channel Simulation

The received QPSK signal constellation before differential decoding is shown in Figure 27 with the frequency offset. As seen in the figure, the center of the constellation points is shifted from their original positions. Figure 28 shows that the differential decoding algorithm successfully placed the scattered points within their previous sectors. Due to the small frequency offset value, the errors were easily corrected. However, for wide ranges of frequency offsets, as the frequency offset value increased, so did the errors.

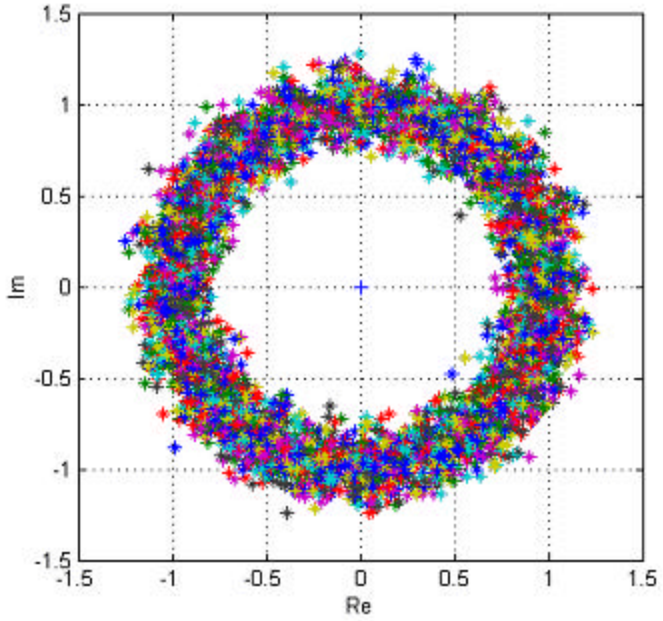


Figure 27. Received Signal Constellation in Channel Model 0 with Frequency Offset of 0.1% of Frequency Spacing (Before Differential Decoding)

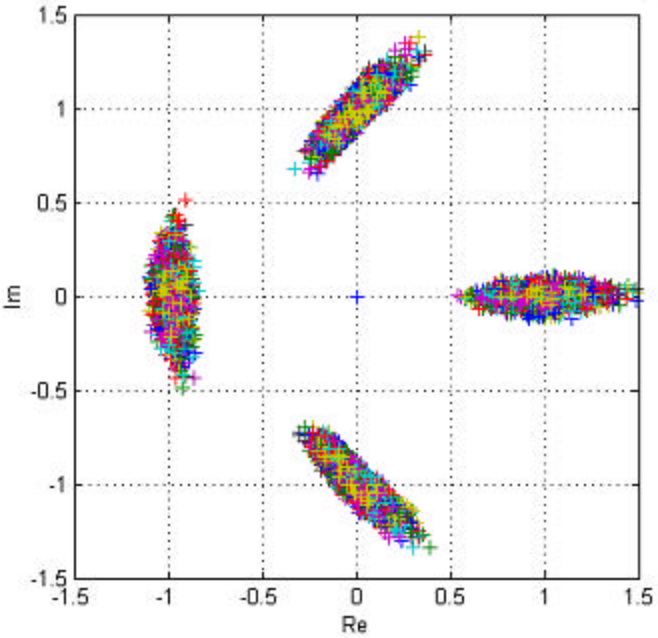


Figure 28. Received Signal Constellation in Channel Model 0 with Frequency Offset of 0.1% of Frequency Spacing (After Differential Decoding)

b. AWGN Channel

This simulation step is important because all theoretical expressions for frequency offset, in the literature, were derived under AWGN channel conditions. Thus, the goal was to study how the system's performance deviated from the theoretical performance with increasing frequency offset.

The OFDM system was configured with the parameters given in Table 10.

# of carriers	# of symbols	Channel model	seed	Interleaver matrix	s range	$\Delta F / R_s$ (%)
48	20000	1	33	[144 139]	0-0.8	0-0.4

Table 10. Frequency Offset in AWGN Channel Simulation

Figure 29 show the BER performance for relative frequency offset values ranging from 0 to 0.4% in 0.1% increments. As can be seen from the figure, above 0.4% frequency offset BER reaches 10^{-1}, which is unacceptably high.

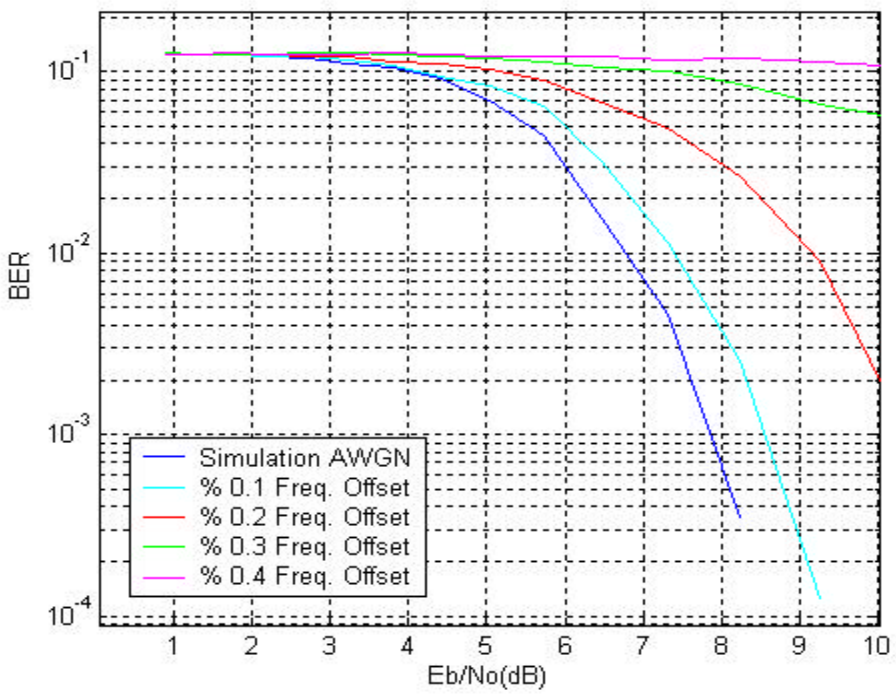


Figure 29. BER versus E_b/N_o Plots for QPSK in AWGN Channel with Relative Frequency Offset Values from 0% to 0.4%

In Figure 29, the required E_b/N_o values increased paraboloically as the relative frequency offset values were increased; we compare these with the BER plot for AWGN simulation. This was an expected result because, by looking at Equation 3.17, the SNR degradation increased with the square of the relative frequency offset.

c. Mobile Channel

The mobile channel models were intentionally chosen to be severe channels to represent the outdoor mobile environments. In this step, the goal was to see the frequency offset effect in a severe mobile channel. Channel 1 is selected for this reason. The system was configured with the same parameters presented in Table 9 but mobile Channel 1 was used for this simulation.

Figure 30 shows the BER plot of the system in mobile channel 1 for relative frequency offset values ranging from 0 to 0.4%.

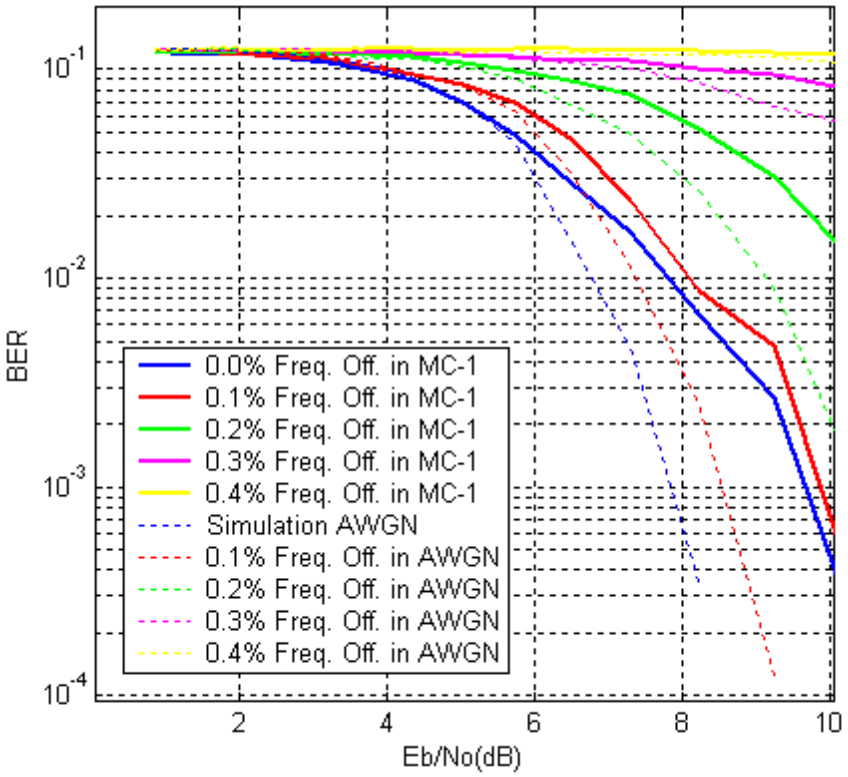


Figure 30. BER versus E_b/N_o Plots for QPSK in Mobile Channel 1 with Relative Frequency Offset Values from 0% to 0.4%

In Figure 30 the dotted lines are from a frequency offset simulation in the AWGN channel and the solid lines are from a frequency offset simulation in mobile channel 1. We can observe the degradation caused by the mobile channel by comparing the dotted and the solid lines and also the OFDM system's performance in mobile Channel 1 can be seen.

5. Performance with Phase Noise

a. Noise-Free Channel

As the first step, the noise-free channel was used to study the phase noise effect exclusively. As mentioned previously, for the phase noise case, $q(t)$ was assumed to be a Wiener process with $E\{q(t)\} = 0$ and $E\left\{\left(q(t_o + t) - q(t_o)\right)^2\right\} = 4\pi\beta|t|$, where β (Hz) is the one-sided 3-dB linewidth of the Lorentzian power density spectrum of the free-running carrier generator. A Wiener process was created in Matlab with zero mean and variance $4\pi\beta|t|$ with β as a percentage of the frequency spacing. The first simulation was run with the configuration parameters given in Table 11. Figure 31 shows the generated phase noise as a Wiener Process.

# of carriers	**# of symbols**	**Channel model**	**seed**	**Interlever matrix**	β / R_s **(%)**
48	8000	0	222	[167 48]	0.01

Table 11. Phase Noise in Noise Free Channel Simulation

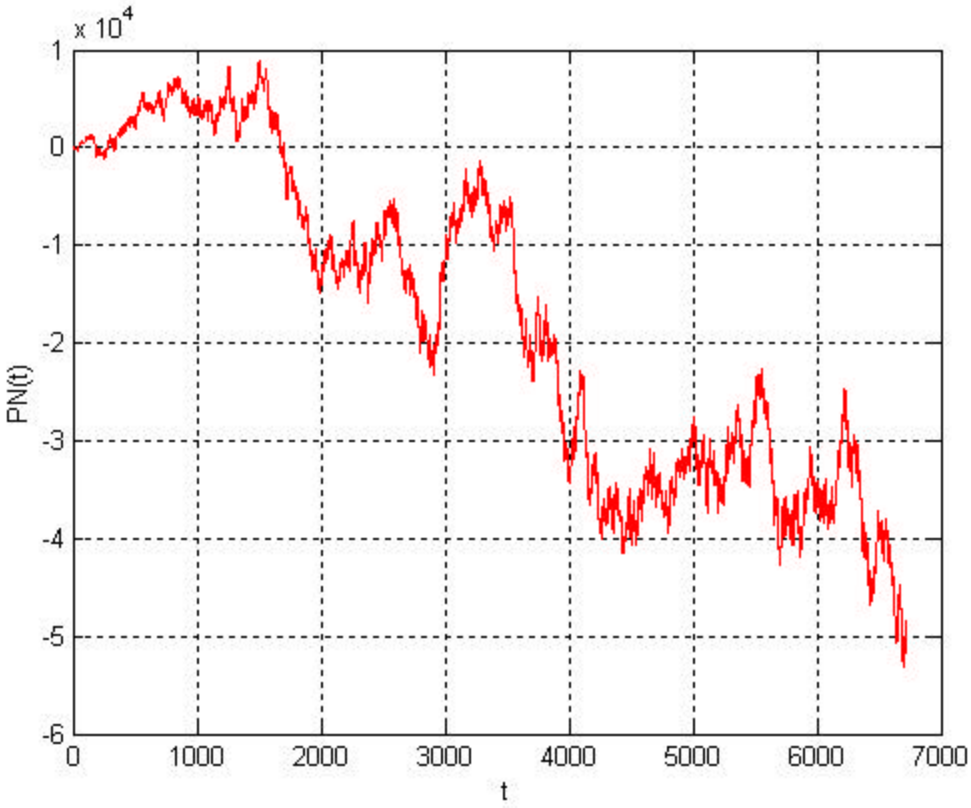


Figure 31. Phase Noise Generated as a Wiener Process

Two different phase-noise sub-blocks were created. The first began the noise generation process with the first OFDM symbol and ended when the last time domain sample of the last OFDM symbol transmitted, and the second created phase noise for each OFDM symbol individually, i.e., one Wiener process for each 80 time domain samples. The first method was used during the simulations since it more accurately reflected the practical operation of a communication system.

Figure 32 shows the received QPSK signal constellation as a result of phase noise, before differential decoding. Figure 33 shows the corrected signal constellation after differential decoding. The differential decoding algorithm successfully placed the scattered points within their respective sectors.

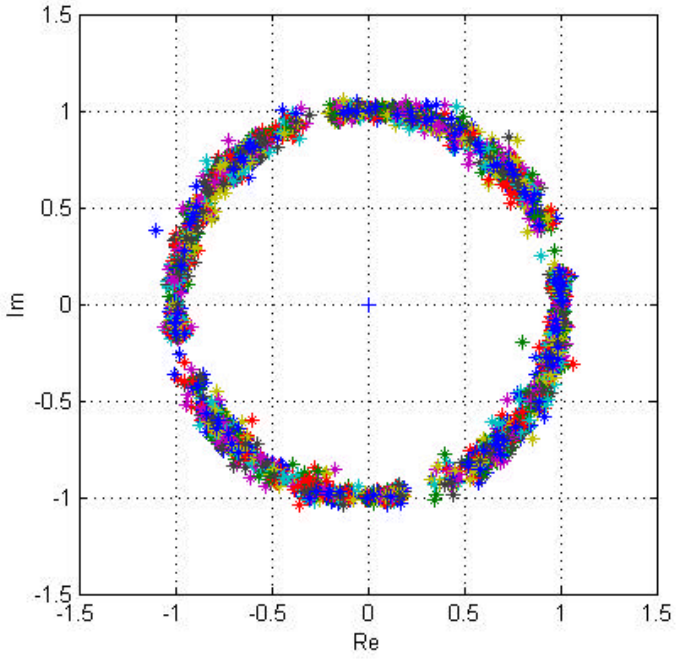


Figure 32. Received Signal Constellation in Channel Model 0 with $b / R_s = 0.0001$. (Before Differential Decoding)

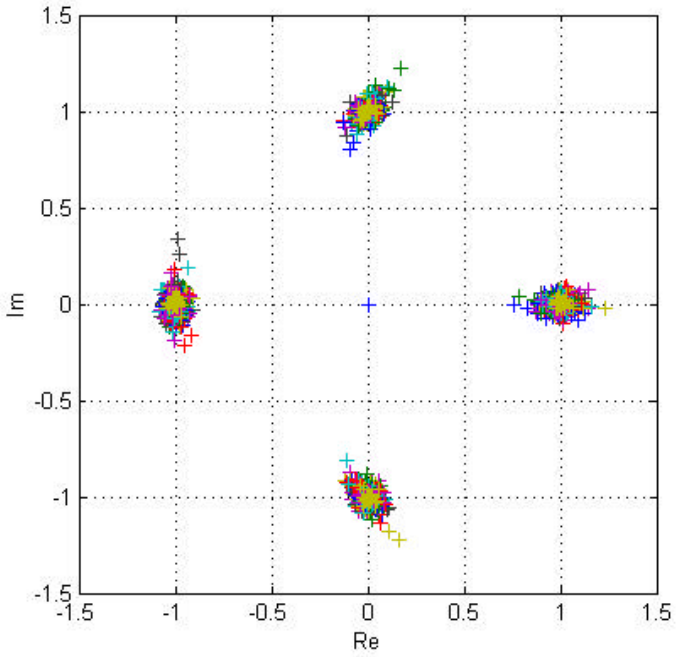


Figure 33. Received Signal Constellation in Channel Model 0 with $b / R_s = 0.0001$. (After Differential Decoding)

b. AWGN Channel

To study the phase noise effect on the system, the code was configured with the parameters in Table 11. Figure 34 shows the BER plots of the system in AWGN channel for normalized bandwidth (b / R_s) values from 0.07% to 0.28% in 0.07% increments.

# of carriers	# of symbols	Channel model	seed	Interleaver matrix	s range	b / R_s (%)
48	20000	1	33	[144 139]	0-0.8	0-0.0028

Table 12. Phase Noise in AWGN Channel Simulation

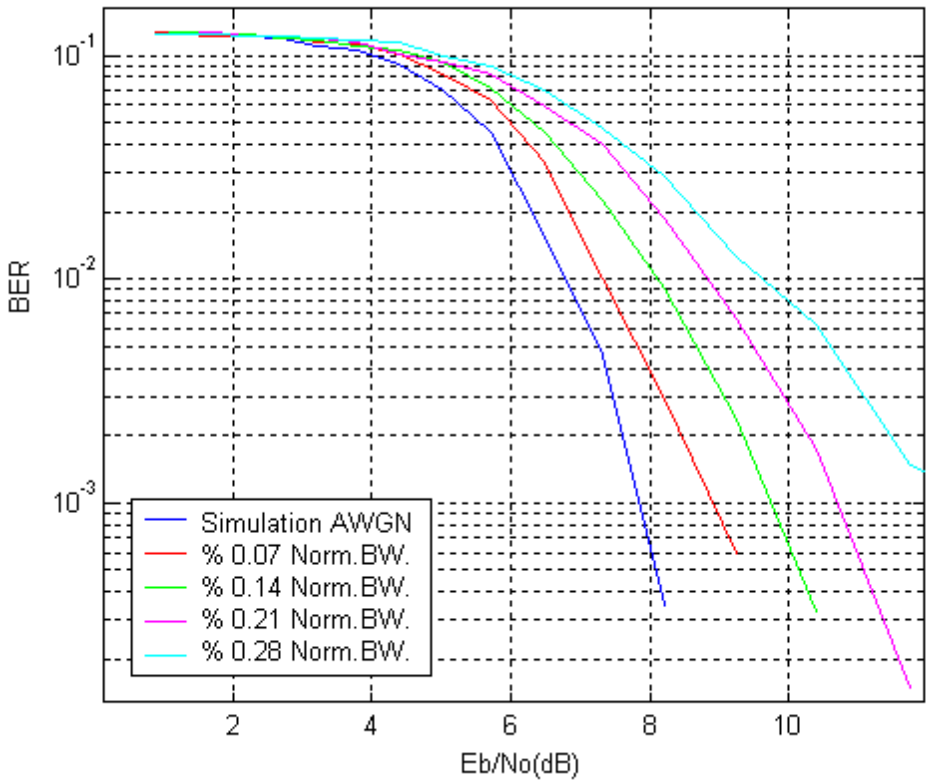


Figure 34. BER versus E_b/N_o Plots for QPSK in AWGN Channel with Normalized Bandwidth b/R_s Values from 0.0% to 0.28%.

Figure 34 suggests a linear increase in the required E_b/N_o values to obtain the same error rates as the normalized bandwidth b/R_s was increased. This was an expected result because from Equation 3.14, the SNR degradation increases linearly with the increasing normalized bandwidth.

c. Mobile Channel

This step was performed to investigate the effects of phase noise in a severe mobile channel. The same parameters in Table 12 were used, but the simulation was performed under mobile Channel 1.

Figure 35 shows the BER performance of the system in mobile Channel 1 for normalized bandwidth values ranging from 0 to 0.28% in 0.07% increments. The dotted lines are for the phase noise simulation in the AWGN channel and the solid lines are for a phase noise simulation in the mobile Channel 1. From Figure 35, we can observe the degradation due to mobile channel by comparing the phase noise simulation results in an AWGN channel (dotted lines) with the phase noise simulation results in the mobile channel (solid lines). The OFDM system under mobile channel conditions requires an E_b/N_o value of approximately 2 dB more than the AWGN case.

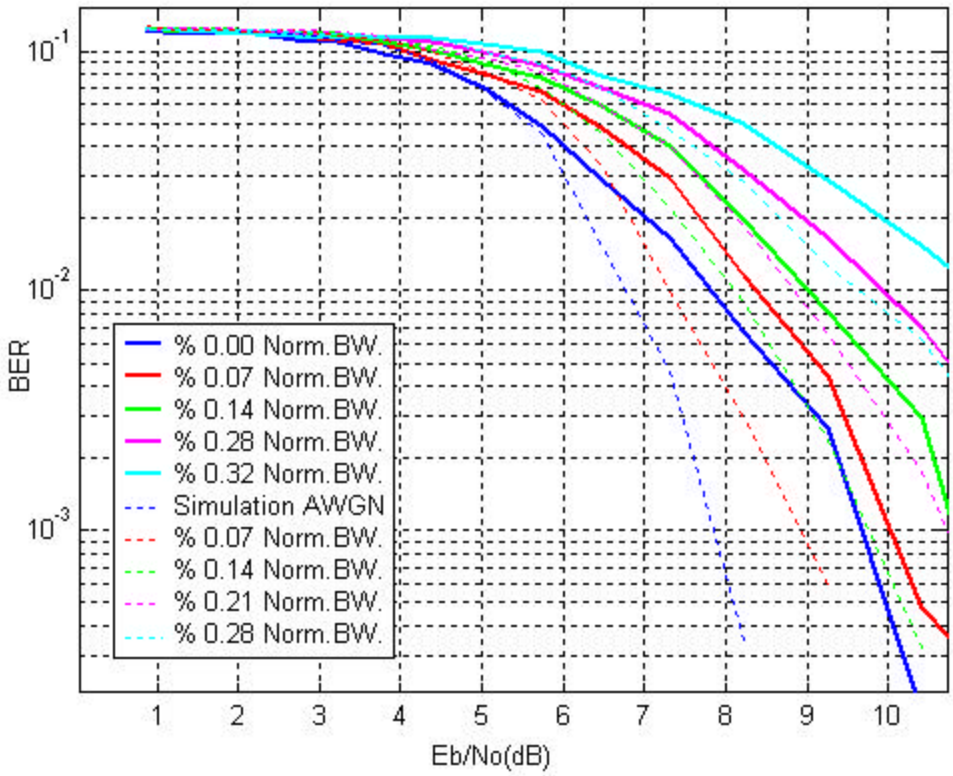


Figure 35. BER versus E_b/N_o Plots for QPSK in Mobile Channel 1 with Normalized Bandwidth b/R_s Values from 0.0% to 0.28%.

C. SUMMARY

This chapter has described a Matlab-based OFDM communication system that can meet the bit rate and performance requirements of the IEEE 802.11a standard. After the verification of the code, several simulation steps were performed under a variety of channel conditions. Some of these simulations especially the ones in steps 4 and 5 took seven hours, which was a limiting factor in the simulation process.

The simulation results did not exactly match the theoretical results; however, they reflected the expected behavior according to the theory. On the other hand, the results were similar to those reported in studies for the same type of simulations.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

The objective of this thesis was to investigate the frequency offset and phase noise effects on OFDM based communication systems. This objective was accomplished by investigating the effects of frequency offset and phase noise on the performance of the system through a comprehensive simulation study.

A. SUMMARY OF THE WORK DONE

A comprehensive background of OFDM based communication systems was presented, including the sub-carrier orthogonality requirement and IFFT/FFT processes. A theoretical analysis of frequency errors and their effects on overall system performance were examined based on the work reported in the literature.

A Matlab simulation of an OFDM communication system was developed. The simulation studies were performed for the following channels: ideal, AWGN, and mobile. The mobile channel includes multipath effects in AWGN. The effects of frequency offset and phase noise were studied for all channel models by observing the deviation from the AWGN case with increasing frequency offset and phase noise. Simulations studied the BER performance plots of OFDM system affected by frequency errors under AWGN and mobile channel conditions.

B. SIGNIFICANT RESULTS AND CONCLUSIONS

The results were satisfactory in terms of the OFDM system exhibiting the expected behavior. However, there were some discrepancies with the theoretical results. These discrepancies stem from many different factors. To make a one-to-one comparison of results from different studies, the simulations must be performed under the same conditions. This was not possible in this thesis because the conditions were either different or not known in detail. Therefore, observing the expected trend, rather than obtaining the exact numerical results, was enough to realize the objectives of this thesis.

From the theoretical analyses given in Chapter 3, we have seen that the SNR degradation due to the Doppler shift is generally negligible. The simulations support this conclusion. In Figure 26, we can see that mobile Channel 1 and 2 have similar BER curves even though the former has 15 Hz and the latter has 5 Hz of Doppler frequency.

The theoretical analyses for frequency offset showed that the OFDM system's performance degrades with the square of the relative frequency offset. This result was observed in our simulations. From Figure 29, a frequency offset of 0.1% required 1 dB more E_b/N_o value than the AWGN case, and a frequency offset of 0.2% required 2.8 dB more E_b/N_o for the same BER.

In the phase noise case, the theoretical analyses showed that the OFDM performance degraded linearly as we increased the normalized bandwidth. The simulation result in Figure 34 supports this point.

Simulation results indicated that mobility causes performance degradation. Figures 30 and 35 show that the performance of OFDM under mobility conditions is worse compared to that of an AWGN channel.

C. SUGGESTIONS FOR FUTURE STUDIES

This thesis was conducted under a certain set of conditions. The effects of changing these conditions on the performance of OFDM systems open a wide range of research topics.

A baseband implementation of the communication system was simulated in this thesis. A future study may focus on an actual RF implementation to observe the system's performance in a more realistic scenario.

Differential encoding/decoding technique was implemented in this thesis. This technique may be replaced with coherent detection techniques to study the system's sensitivity to synchronization issues including the use of pilot tones. This study may help evaluate if the added complexity of the coherent techniques pays off in terms of increased performance.

This work focused on the effects of frequency errors and studied only four multipath channel models. A more comprehensive study on the effects of mobile channels may be considered in a future work. Such a study will help in the adoption of OFDM for the fourth-generation mobile communications.

QPSK was used in this thesis. The effects of other signaling schemes on the system's performance may be studied to find an optimum signaling scheme. Also of interest would be a study of the effects of changing the OFDM system parameters used in this thesis, especially the guard interval and the number of OFDM sub-carriers on the performance of the system.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB CODE EXPLANATION

This appendix explains the function of each m-file and their relations with each other. The code is separated into three parts: transmitter, channel and receiver.

COFDMSIM: This m-file is the main simulation-governing file. The main configuration parameters, such as sub-carrier number, FFT point number, noise variance range, multi-path losses, multi-path delays, Doppler frequencies, guard interval length, channel models, seed value, interleaver matrix dimensions, time/frequency differential encoding and constellation sizes are chosen here. After the system is configured with these parameters, the m-file CHANCDL is called.

CHANCDL: This m-file performs the simulations depending on the channel model chosen in COFDMSIM. This m-file calls CDRCDLFT.

A. TRANSMITTER

1. CDRCDLFT

This is the COFDM encoder with CDL interleaver and is called to compute the frequency array of prearranged modulation values and the matrix of differentially encoded complex values. It calls MLTPL.

MLTPL computes the possible interleaver matrix dimensions. CDRCDLFT creates an error message if the interleaver matrix cannot accommodate all the created symbols.

MARYMSG generates the required number of bits randomly and then sends them to the convolutional encoder. MARYMSG calls CNV_ENCD and BM.

CNV_ENCD convolutionaly encodes the randomly generated bits in MARYMSG. *BM* then converts the convolutionaly-encoded bits to m-ary symbols. (*m* value was chosen in COFDM.)

As the last step, MARYMSG creates the *m-ary msg* matrix, which contains the message symbols. After obtaining the message symbols from MARYMSG, CDRCDLFT calls CDLILV, which interleaves the message symbols. During the interleaving operation, ROTM is called to rotate the input vector.

With the creation of the interleaved message matrix, CDRCDLFT calls BM again and creates n-ary message matrix that are to be transmitted. (The choice of n was also made in COFDM and it determines the constellation. In the simulations, $n = 2$ for QPSK signaling was used.)

DIFCDRFT performs time or frequency differential encoding, as required. If time differential encoding is chosen, a cumulative summation on each column is executed. For frequency differential encoding, a cumulative summation on each row is performed. The cumulative summation is executed such that the first element is added to the next element and replaces the second element in the current summation, then adds the current sum to the third element and replaces it also with the current sum. All the sums are modulo-N.

After deriving the differentially encoded matrix, its elements are encoded as N-ary complex modulation values. The unit circle is divided into the number phase partitions depending on the signaling constellation and each element of the differentially encoded matrix is mapped to one of these phases. A reference row of ones is appended to the message array to provide a reference starting point for the receiver, in the case of time differential encoding, and two reference columns of ones are inserted in the case of frequency differential encoding. These reference symbols are stripped off in the receiver during differential decoding.

CMV2FA is the last m-file CDRCDLFT calls.

CMV2FA rearranges the differentially encoded complex modulation values into a frequency array to prepare them for OFDM frequency generation by the IFFT.

This final step of the encoder block has the convolutionaly encoded, block interleaved, differentially encoded and prearranged frequency array. The frequency array is returned to the CHANCDL m-file for the IFFT process. TDA is the m-file responsible for the IFFT process.

2. TDA

Performs the IFFT processing and generates the OFDM frequencies. The transmitted OFDM symbols for multi-path effects are prepared by appending a periodic guard interval with length *Ng*. (*Ng* is chosen in COFDM and, guard interval details were discussed in Chapter II.)

The time domain samples representing the OFDM symbol appended with the guard interval are returned to CHANCDL as the signal to be transmitted through the channel. Figure 36 shows the relations of m-files in the transmitter block diagram.

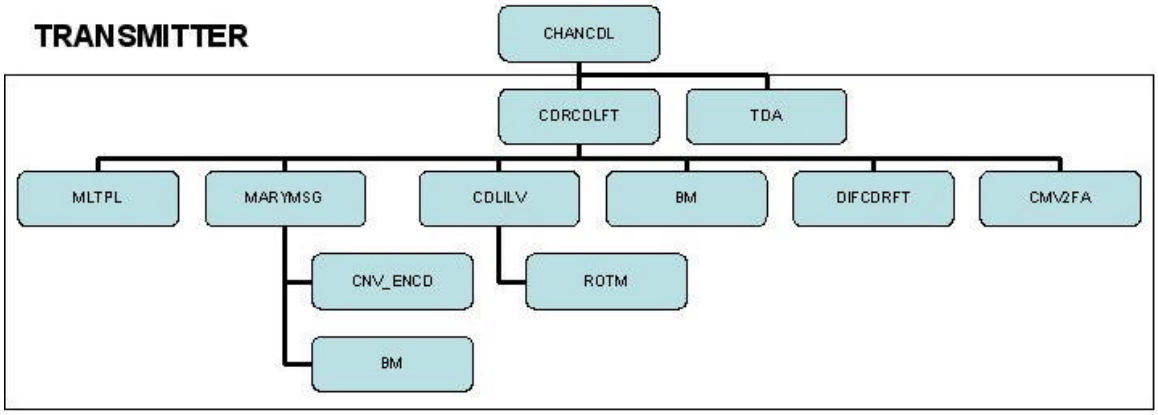


Figure 36. Hierarchical m-file Block Diagram for the Transmitter

B. CHANNEL

One of the three main channel models can be chosen and combined with frequency offset or phase noise. The phase noise can be applied prior to the channel to simulate the phase noise case.

1. Phase Noise Simulations

The time domain OFDM symbol samples from the TDA m-file are taken and sent to the phase noise m-files of PHSNS or phsns2.

phsns2 generates phase noise as a Wiener process and then applies it to all the samples that are being transmitted. Then, CHANCDL sends the time domain samples to the desired channel model. Figure 37 illustrates the place of the phase noise block in the m-file hierarchy.

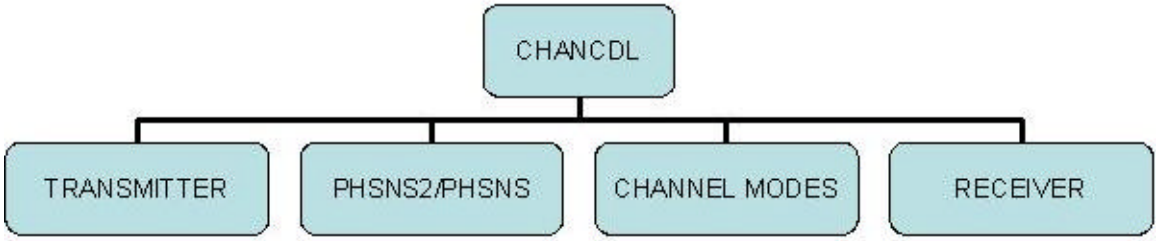


Figure 37. Phase Noise Creating Block in the m-file Hierarchy

2. Channel Model Simulations

Time domain samples are next sent to the channel model of choice.

a. Channel Model 0

This is the ideal noise-free channel used for code check simulation purposes. Here, the output of the transmitter block is sent directly to the receiver. This model can also be used to check whether frequency offset or phase noise can be implemented. These two possibilities are illustrated in Figure 38.

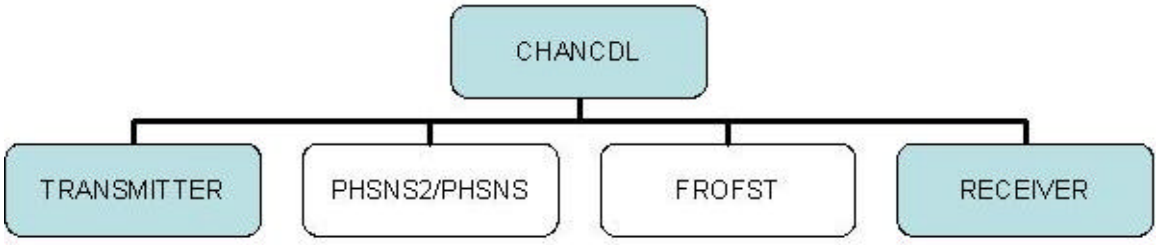


Figure 38. Channel Model 0 m-file Hierarchy

b. Channel Model 1

If channel model 1 is chosen, then CHANCDL directs the time domain samples to AWGN.

AWGN generates a matrix of complex random numbers chosen from a normal distribution with mean 0 and variance 1. This matrix has the same dimensions as the time domain samples matrix. This matrix is then multiplied by the noise variance parameter chosen in COFDM and added to the time domain signal matrix. This channel model can either be used individually or as a component of a mobile channel.

c. Mobile Channel Models

Four different mobile channels are created by combining an AWGN channel with multi-path effects. If desired, the multi-path channels can also be used individually for simulation purposes. The choice of a mobile channel model is determined in the COFDM m-file.

Mobile Channel 1 and 2: If one of these mobile channels is chosen in COFDM, then CHANCDL sends the time domain signal to CHUHF. *CHUHF* functions as the multi-path component of mobile Channels 1 and 2. It calls DLINE to realize a delay line and set up the multiple delayed paths.

CVDD creates a continuous digital delay element in which the time domain samples are filtered using an eight-tap FIR filter whose tap coefficients are a function of the desired delay.

After receiving the delayed paths from DLINE, CHUHF next calls OFST, which applies a frequency offset as a ratio of the max Doppler shift to the direct path of time domain samples.

RAY_DOP calculates the max Doppler shift frequency as a ratio of frequency spacing. A sequence of complex numbers with zero mean and variance 0.5 is created having a Rayleigh envelope with a mean square value of 1. The real and imaginary parts are generated independently.

Finally, CHUHF includes the power losses for the individual multi-paths by multiplying each loss amount by the respective delay line output arrays. At this point, CHANCDL has the time domain representation of the transmitted signal plus the multi-path effects. The time domain samples are then sent to AWGN as the second component of the mobile channel.

Mobile Channel 3: If this model is chosen in COFDM, then CHANCDL routes the signal to channel_a, which models different Doppler frequencies, power losses, envelope (Ricean) and delays. It calls *jakes*, which implements FIR filtering by using a Hamming window.

Mobile Channel 4: If this model is chosen, CHANCDL sends our signal to channel_b, which is the last mobile channel model with very similar characteristics to model 3. It also calls *jakes* for filtering purposes.

Mobile Channels 3 and 4 also calls AWGN as the second component of the mobile channel.

Figure 39 shows hierarchical architecture of all the mobile channels. For each simulation, it is only possible to choose one mobile channel. However, as the white blocks suggests, it is possible to combine mobile channels with phase noise or frequency offset. Mobile channels can also be used individually.

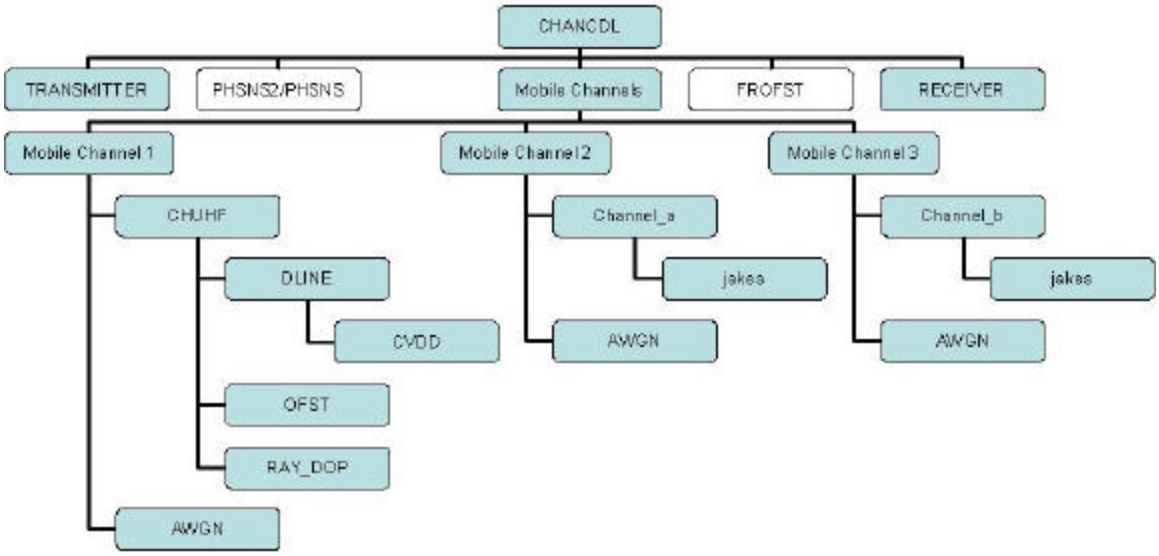


Figure 39. Mobile Channels m-file Hierarchy

Figure 40 summarizes the entire channel model m-file architecture.

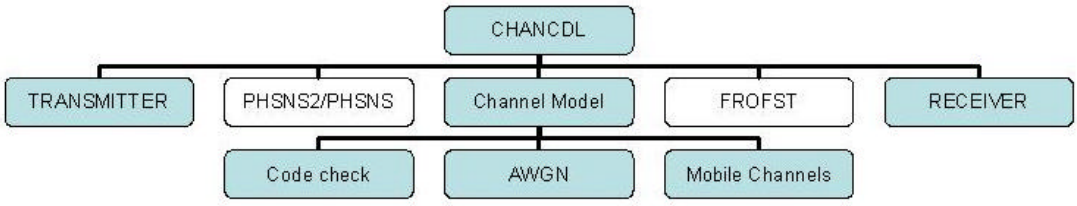


Figure 40. Summary of Channel Model m-file Architecture

3. Frequency Offset Simulations

If frequency offset simulations are desired, CHANCDL directs the time domain samples to frofst, after the channel model of choice.

frofst applies the frequency offset to each of the transmitted time domain samples as a percentage of the frequency spacing. Figure 41 shows the place of frequency offset generating block in the m-file hierarchy.

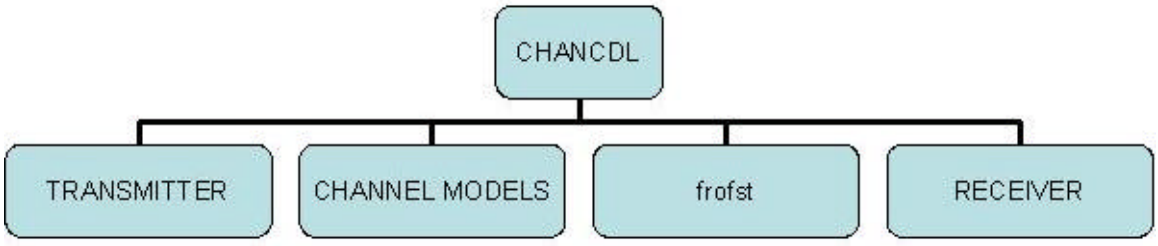


Figure 41. Frequency Offset Block in the m-file Hierarchy

Finally, all the m-files in the channel part have been discussed .The last part of the code implements the receiver.

C. RECEIVER

The first m-file in the receiver part is the ITDA.

ITDA takes the time domain samples, removes the guard interval and performs FFT to create the frequency domain samples. The decoding functions are performed within the DECDRCDL.

DECDRCDL performs the decoding and deinterleaving functions. It calls FA2CMA, which reconstructs the frequency array as a complex modulation array with the correct ordering of frequencies.DECDRCDL next calls DFDCDRFT to decode the complex modulation values differentially.

DFDCDRFT takes the complex modulation values from *FA2CMA* and performs inverse mapping of the complex numbers to *N*-ary symbols based on the value chosen in COFDM. Differential decoding is the inverse of encoding performed in the transmitter.

After receiving *N*-ary message from DFDCDRFT, DECDRCDL calls MB and BM respectively to reformat the received symbols into *m*-ary symbols.

The next step in the receiver block is to deinterleave the message symbol array. DECDRCDL calls CDLDLV, which creates the deinterleaved message matrix by using the ROTM as described in the transmitter block.

These deinterleaved message symbols are next changed to binary and sent to the Viterbi decoder.

VITERBI performs the Viterbi decoding and during the process it calls BIN2DECI and DECI2BIN m-files.

Finally, DECDRCDL takes the received bit sequence from VITERBI, reformats it as an *m*-ary message sequence and creates the received message matrix. Figure 42 shows the receiver m-file block diagram.

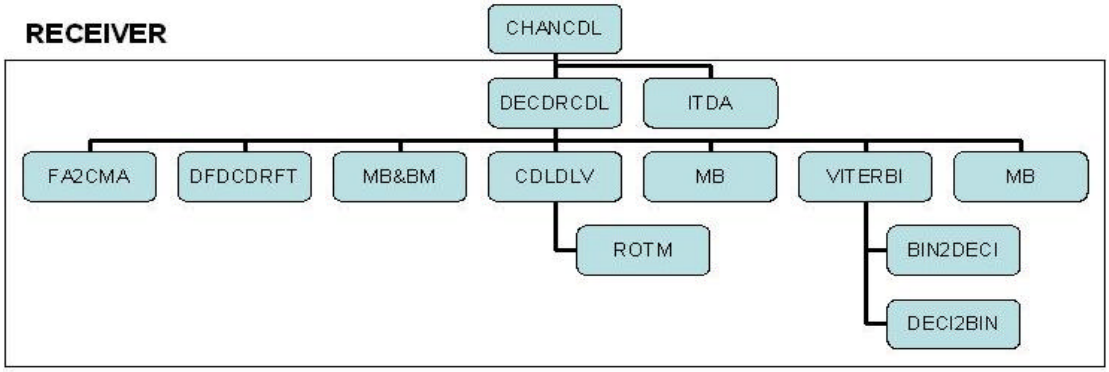


Figure 42. Hierarchical m-file Block Diagram for the Receiver

At this step, CHANCDL successfully governed the transmitter and receiver operations. Further steps are taken to plot the necessary data to be used for analysis purposes. The simulations end after providing the simulations' performance curves.

Figure 43 summarizes the entire m-file algorithm.

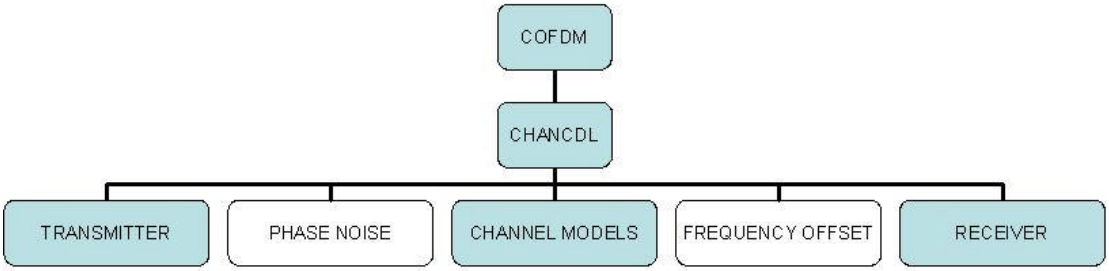


Figure 43. General m-file Block Diagram

APPENDIX B. MATLAB CODE

In this appendix, the m-files used in the simulations are presented. Descriptions of the m-files included here can be found in Appendix A.

```
%AWGN
%--------------------------------------------------------------------------------
%Title       : Additive White Gaussian Noise (Channel Model 1)
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%--------------------------------------------------------------------------------
%INPUTS :
% X     : Time domain samples of the transmitted signal
% s     : Seed parameter
% sigma : AWGN Noise variance parameter for calculating Eb/No
%
%OUTPUTS:
% Y     : Time domain samples of the transmitted signal plus AWGN
%--------------------------------------------------------------------------------
function Y = awgn(X,sigma,s)
%
%Find dimensions of the input array
[rr,cc]=size(X);
%
randn('seed',s+30);
%Generate a random real part
wreal=randn(rr,cc);
%Generate a random imaginary part
randn('seed',s+40);
wimg=i*randn(rr,cc);
%
%An array of random complex entries chosen from a normal distribution with
%mean 0.0 and variance 1.0. Array dimensions are the same as X.
W=wreal+wimg;
%
%Random noise multiplied by the sigma factor and added to the signal.
Y=X+(sigma.*W);
```

```
%BIN2DECI
%-------------------------------------------------------------------------------
%
%Title       : Binary To Decimal Conversion
%Author      : Tan Kok Chye, Naval Postgraduate School
%-------------------------------------------------------------------------------
%INPUT :
% v_x  : Binary input
%
%OUTPUT:
% v_y  : Decimal output
%-------------------------------------------------------------------------------
function v_y=bin2deci(v_x)
v_l=length(v_x);
v_y=(v_l-1:-1:0);
v_y=2.^v_y;
v_y=v_x*v_y';
```

```
%BM
%--------------------------------------------------------------------------------
%
%Title       : Binary to M-ary Converter
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
%INPUTS:
% q : Base 2 exponent for M-ary symbol generation
% v : Binary data vector
%
%OUTPUTS:
% m : M-ary output vector in decimal notation
%--------------------------------------------------------------------------------
function m=bm(q,v)
%
%Find the length of input vector,v,and determine if there is a remainder
%after dividing by q
n=length(v);
r=rem(n,q);
%
%If there is no remainder,don't pad v input vector. Otherwise add the appropriate
%number of zeros to generate a code word with an exact multiple of q bits.
%
if r==0
v=v;
else
v=[v zeros(1,q-r)];
end
%
%Place least significant bit of the symbol on the left end.
map=1;
for j=1:q-1
map=[map 2^j];
end
%
%Remove q bits at a time from v to generate m-ary symbol values.
n=length(v);
p=round(n/q);
A=zeros(q,p);

A(:)=v;
m=map*A;
m_ary_msg=m;
```

```
%CDLDLV
%-------------------------------------------------------------------------------
%
%Title       : CDL Block Deinterleaver
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
% l    : Intermediate matrix row number
% k    : Intermediate matrix column number
% dcase: Indicates the deinterleaving method to be
% used (9 different cases). In our thesis only block interleaving was used.
% si   : Input message string to be deinterleaved
% SYNC : Frame synchronization bits (Not used in COFDM simulation)
%
%OUTPUTS:
% s : Interleaved output string
%-------------------------------------------------------------------------------
function s=cdldlv(l,k,dcase,si,SYNC)
si(length(si)+l-length(SYNC):length(si))=zeros(l,length(SYNC));
N=length(si);
if l*k==N
x=zeros(l,k);
x(:)=si;
K=(1:k)-1;
CR=K.*(K+1)/2;
L=(1:l)-1;
RR=L.*(L+1)/2;
%
   if dcase==1
      for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
      end
   elseif dcase==2
      for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
      end
   elseif dcase==3
      for kk=1:l
      x(kk,:)=rotm(x(kk,:),RR(kk));
      end
   elseif dcase==4
      for kk=1:l
      [z,x(kk,:)]=rotm(x(kk,:),RR(kk));
```

```
        end
    elseif dcase==5
        for kk=1:k
        x(:,kk)=rotm(x(:,kk),CR(kk));
        end
        for ll=1:l
        x(ll,:)=rotm(x(ll,:),RR(ll));
        end
    elseif dcase==6
        for kk=1:k
        [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
        end
        for ll=1:l
        x(ll,:)=rotm(x(ll,:),RR(ll));
        end
    elseif dcase==7
        for kk=1:k
        x(:,kk)=rotm(x(:,kk),CR(kk));
        end
        for ll=1:l
        [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
        end
    elseif dcase==8
        for kk=1:k
        [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
        end
        for ll=1:l
        [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
        end
    end
x=x';
s=x(:);
s=s';
end
```

```
%CDLILV
%-------------------------------------------------------------------------------
%
%Title       : CDL Block Interleaver
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
% l    : Intermediate matrix row number
% k    : Intermediate matrix column number
% dcase: Indicates the deinterleaving method to be
% used (9 different cases). In our thesis only block interleaving was used.
% si   : Input message string to be deinterleaved
% SYNC : Frame synchronization bits (Not used in COFDM simulation)
%
%OUTPUTS:
% si : Interleaved output string
%
%Subroutines Used : rotm.m
%-------------------------------------------------------------------------------
function si = cdlilv(l,k,dcase,s,SYNC)
N=length(s);
if l*k==N
x=zeros(l,k);
x=x';
x(:)=s;
x=x';
Intermediate_mx=x;
K=(1:k)-1;
CR=K.*(K+1)/2;
L=(1:l)-1;
RR=L.*(L+1)/2;
%
   if dcase==1
      for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
      end
   elseif dcase==2
      for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
      end
   elseif dcase==3
      for kk=1:l
      [z,x(kk,:)]=rotm(x(kk,:),RR(kk));
```

```
        end
    elseif dcase==4
        for kk=1:l
        x(kk,:)=rotm(x(kk,:),RR(kk));
        end
    elseif dcase==5
        for ll=1:l
        [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
        end
        for kk=1:k
        [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
        end
    elseif dcase==6
        for ll=1:l
        [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
        end
        for kk=1:k
        x(:,kk)=rotm(x(:,kk),CR(kk));
        end
    elseif dcase==7
        for ll=1:l
        x(ll,:)=rotm(x(ll,:),RR(ll));
        end
        for kk=1:k
        [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
        end
    elseif dcase==8
        for ll=1:l
        x(ll,:)=rotm(x(ll,:),RR(ll));
        end
        for kk=1:k
        x(:,kk)=rotm(x(:,kk),CR(kk));
        end
    end
x;
si=x(:);
si=si';
end
si(length(si)-length(SYNC)+1:length(si))=SYNC;
```

```
%CDRCDLFT
%-------------------------------------------------------------------------------
%
%Title       : COFDM Encoder with CDL Interleaver
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
% pic    : Parameter indicating the figure number
% s      : Seed parameter
% freqno : Number of OFDM frequencies (sub-carriers) used in each message array
% rintlv : Parameter for intermediate matrix row number
% cintlv : Parameter for intermediate matrix column number
% N      : Number of FFT frequency sample points,must be larger than freqno
% mary   : Initial M-ary symbol format (OFDM symbol bit length)
% nary   : Final N-ary symbol format (PSK symbol bit length)
% fort   : Selects either frequency (fort=1) or time (fort=0) differential encoding
%
%OUTPUTS:
% Fa : Frequency array of prearranged modulation values
% MD : Matrix of differentially encoded complex values
%(unity magnitude and one of N-ary possible phases (N-PSK))
% B  : Matrix of 8-ary symbols
% nsymno: Number of N-ary generated symbols
%
%Subroutines Used : marymsg.m,cdlilv.m,mb.m,bm.m,difcdrft.m,cmv2fa.m
%-------------------------------------------------------------------------------
func-
tion[Fa,MD,B_ce,B_random,nsymno]=cdrcdlft(pic,dcase,s,freqno,rintlv,cintlv,N,mary,na
ry,fort);
%
% Determine whether the number of OFDM frequencies are even, indicated
% by the "freqno" parameter. If odd, go to error message. Odd frequencies are not al-
lowed
% since the formation of the frequency array is symmetrical and even.
%
if rem(freqno,2)~=0
disp('ERROR: The number of matrix columns, representing OFDM frequencies, must be
an even number!')
elseif rem(freqno,2)==0
%
% Check if interleaver matrix dimensions are greater than freqno.
% If not, then display error message and stop.
%
```

```
   if (rintlv*cintlv)<(freqno)
   disp('')
   disp('ERROR: The row and column interleave parameters are not compatible with # of
OFDM frequencies!')
   disp('')
   else
% Calculate the row symbol number
   symno=rintlv*cintlv/freqno;
%
% Display error message if symno and freqno not compatible with rintlv and cintlv and
stop.
% If not compatible,the interleaver function does not work correctly.
%
      if rem(symno,1)~=0
      disp('')
      disp('ERROR: The row and column interleave parameters are not compatible with #
of OFDM frequencies!')
      disp(' For the entered rintlv, cintlv, and freqno parameters, the calculated symno is:')
      disp(symno)
      multiesall=mltpl(rintlv,cintlv);
      multies=multiesall(1,(2:length(multiesall)-1));
      disp(' Possible choices for freqno based upon rintlv and cintlv are:')
      disp('')
      disp(multies)
      elseif rem(symno,1)==0
         if freqno >= N;
         disp('')
         disp('ERROR: The number of frequency points, N, needs to be increased !')
         disp('N must be larger than:')
         disp('')
         disp(freqno)
         disp('')
         elseif freqno < N;
%
% Generate a random message matrix of m-ary symbols,based upon parameter,mary.
         Nmbr_of_symbols =symno*freqno;
         [B_ce,B_random]=marymsg(mary,symno,freqno);
         Rndm_m_ary_msg=B_random;
%
% Perform a block interleaving function on the matrix, B, with rintlv rows
% and cintlv columns.
         SYNC=[];
         [Br Bc]=size(B_ce);
         Bt=B_ce';
         Bvect=Bt(:)';
```

```
        si=cdlilv(rintlv,cintlv,dcase,Bvect,SYNC);
        Bi=reshape(si,Bc,Br)';
        Intrlvd_array=Bi;
        m1=bm(nary,mb(mary,Bi));
        lengthm1=length(m1);
        nsymno=lengthm1;
        remm1=rem(lengthm1,freqno);
           if remm1==0;
           m1=m1;
           else
           zero=zeros(freqno-remm1);
           m1=[m1 zero(1,:)];
           end
        length2m1=length(m1);
        m=(reshape(m1,freqno,length2m1/freqno))';
        N_ary_msg=m;
%
% Generate a differentially encoded matrix of complex
% values with unity magnitude and one of (2^n) equal phases.
      MDD=difcdrft(nary,m,fort);
      [MDm MDn]=size(MDD);
      MD=MDD;
      Cmplx_mod_array=MDD;
%
% Form the frequency array of modulation values that include guard interval.
      Fa=cmv2fa(N,MD);
      Freq_array=Fa;
         end
      end
   end
end
```

```
%CHANCDL
%-------------------------------------------------------------------------------
%
%Title       : Simulations for AWGN & Multipath Fading Channel
%Author      : Dave Roderick, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%-------------------------------------------------------------------------------
%Subroutines Used : cdrcdlft.m,tda.m,awgn.m,chuhf.m,itda.m,decdrcdl.m,check.m
%-------------------------------------------------------------------------------
function
chancdl(chnmdl,wait,pic,dcase,s,freqno,rintlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss
,dly,dop,freqspace,fort)
sigvect=sigs;
klgth=length(k);
chklp=1;
symno=rintlv*cintlv/freqno;

%%%%FOR PHASE NOISE%%%%%
%for perctB=0:.0008:.0032

%%%%FOR FREQUENCY OFFSET%%%%
%for doner=0:0.0007:.0028

errvect=[];
bervect=[];
freqerrmx=[];
errsperpr=[];
Es_No=[];
Eb_No=[];
sermx=[];
bermx=[];
rowerrmx=[];

 for lp=1:length(sigvect);

[xmt,modvals,B_ce,B_random,nsymno]=cdrcdlft(pic,dcase,s,freqno,rintlv,cintlv,N,mary,
nary,fort);
   xmtifft=tda(Ng,xmt);

%%%%%%%%%%%FOR PHASE NOISE%%%%
   %sandnaphns=phsns2(xmtifft,freqspace,perctB);
   %xmtifft=sandnaphns;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   xmtpts=1:length(xmtifft);
```

```
    if chnmdl==0
    sandn=xmtifft;

    elseif chnmdl==1
    disp(['Sigma=',num2str(sigvect(lp))]);
    sandn=awgn(xmtifft,sigvect(lp),s);

    elseif chnmdl==21
    sandmltpth=chuhf(s+1,xmtifft,loss,dly,dop,N,freqspace);
    disp(['Sigma=',num2str(sigvect(lp))]);
    sandn=awgn(sandmltpth,sigvect(lp),s);

    elseif chnmdl==22
    sandmltpth=chuhf(s+1,xmtifft,loss,dly,dop,N,freqspace);
    disp(['Sigma=',num2str(sigvect(lp))]);
    sandn=awgn(sandmltpth,sigvect(lp),s);

    elseif chnmdl==23
    disp(['Sigma=',num2str(sigvect(lp))]);
    sandmltpth=channel_a(xmtifft);
    sandn=awgn(sandmltpth,sigvect(lp),s);

    elseif chnmdl==24
    disp(['Sigma=',num2str(sigvect(lp))]);
    sandmltpth=channel_b(xmtifft);
    sandn=awgn(sandmltpth,sigvect(lp),s);
    end

%%%%%%%%FOR FREQUENCY OFFSET%%%%%%
    %sandnofst=frofst(sandn,doner);
    %sandn=sandnofst;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    sandnfft=itda(Ng,sandn);
    K=(length(modvals(1,:)))/2;

[rcvd,rcvd_bit,random_msg,random_bit,M,MM]=decdrcdl(pic,dcase,K,sandnfft,nsymno,
freqno,rintlv,cintlv,mary,nary,fort,B_random);
    %%Transmitted_msg=B_random;
    Transmitted_msg=random_msg;
    Received_msg=rcvd;

    [er-
rors,bit_error,freqerrs,errmx,rowerrs]=check(pic,random_msg,random_bit,rcvd,rcvd_bit,
n,k(chklp),blklgth);
```

```
    errvect=[errvect,errors];
    bervect=[bervect,bit_error];
    freqerrmx=[freqerrmx;freqerrs];
    rowerrmx=[rowerrmx;rowerrs];

    crntEs_No=1/(2*N*(sigvect(lp)^2));
    crntEb_No=crntEs_No;
    % %based on M=4 i.e. for coded QPSK, Eb=Es.
    Es_No=[Es_No,crntEs_No];
    Eb_No=[Eb_No,crntEb_No];
    Es_Nodb=10*log10(Es_No);
    Eb_Nodb=10*log10(Eb_No);
 end

ser=errvect/(symno*freqno);
ber=bervect/(2*symno*freqno);
sermx=[sermx;ser];
bermx=[bermx;ber];
errsum=sum(errvect);
errsperpr=[errsperpr,errsum];
errmax=max(rowerrmx');
%
% PLOTS
%
figure(pic+1)
plot(modvals,'*')
hold on;
plot(0,0,'+')
hold off;
%title(['Transmitted Signal Constellation',int2str(2^nary), '-ary (QPSK) '])
xlabel('Re');
ylabel('Im');
axis('square');orient tall;grid
pause(wait);
%%%%Plot of FFT POINTS%%%%
%figure(pic+2)
%plot([0:N-1],abs(xmt),'*')
%title(['Frequency Array Plot (number of FFT frequency points are ',int2str(N),')'])
%xlabel(['Guard interval length is ',int2str(N-freqno)])
%axis('square');orient tall;grid
%pause(wait);
%
figure(pic+2)
surf(abs(modvals));
shading interp;grid;orient tall
```

```
%title(['Magnitude Plot of Transmitted Signal'])
xlabel('OFDM Subcarrier Number')
ylabel('Symbol Row Number')
zlabel('Signal Magnitude')
pause(wait);
%
figure(pic+3)
plot(M,'*');hold on;
plot(0,0,'+');hold off;
%title(['Received Signal Constellation', int2str(2^nary),'-ary (QPSK), (Before Differential
Decoding)'])
xlabel('Re');
ylabel('Im');
orient tall;axis('square');grid
pause(wait);
%
figure(pic+4)
plot(MM,'+')
hold on;
plot(0,0,'+')
hold off;
%title(['Received Signal Constellation',int2str(2^nary),'-ary (QPSK), (After Differential
Decoding)'])
xlabel('Re');
ylabel('Im');
orient tall;axis('square');grid
pause(wait);
%
figure(pic+5)
surf(abs(M));
shading interp
grid;orient tall
%title(['Magnitude Variation of Received Signal (Sigma=',num2str(sigvect(lp)),')'])
xlabel('OFDM Subcarrier Number')
ylabel('Symbol Row Number')
zlabel('Signal Magnitude')
pause(wait);
%
 if errsum~=0
%%%%PERFORMANCE PLOTS%%%%

%theoretical on top of simulation curves
%q=length(Eb_No)
   %for i=1:q
   %P=0.5*(erfc(Eb_No(1,i))*(1-0.25*erfc(Eb_No(1,i))));
```

```
    %Pb(1,i)=P;
    %end
    figure(pic+6)
    %semilogy(Eb_Nodb,Pb,'m');grid;hold on;
    semilogy(Eb_Nodb,ber);hold on;grid
    if fort==1
      %title('Performance curve: BER vs. Eb/No for QPSK Signaling (Freq. Diff.Enc.)')
    elseif fort==0
      %title('Performance curve: BER vs. Eb/No for QPSK Signaling (Time. Diff.Enc.)')
    end
    xlabel('Eb/No(dB)');
    ylabel('BER');
    %legend('theorical curve','simulation curve');orient tall
%
    figure(pic+7)
    %semilogy(Es_Nodb,2*Pb,'m');hold on
    semilogy(Es_Nodb,ser);grid;hold on
    if fort==1
      %title('Performance curve: SER vs. Es/No (Freq. Diff.Enc.)')
    elseif fort==0
      %title('Performance curve: SER vs. Es/No (Time. Diff.Enc.)')
    end
    xlabel('Es/No(dB)');
    ylabel('SER');
    %legend('theoretical curve','simulation curve');orient tall
 end
end
```

```
%channel_a
%-------------------------------------------------------------------------------
%
%Title       : Mobile Channel 2
%Author      : Prof.Roberto Cristi, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%-------------------------------------------------------------------------------
%Subroutines Used : jakes.m
%-------------------------------------------------------------------------------
function  yr=channel_a(xt)
[asd asn]=size(xt);
xt=xt.';
xt=xt(:).';
%
%  yr=channel(xt)
%  with xt=transmitted vector
%       yr=received vector
%       length(yr)=length(xt)
%
% if the program gives you an error, just increase the size of the
% transmitted vector "xt"

Fs=1.25*10^6;           % sampling frequency (Hz)
Fd=200;                 % doppler frequency (Hz)

%Parameters for Channel A
Td=[0.0, 0.25,0.5,1.0,1.9,2.2];                 % time delays (in microsec)
PdB=[0.0,-1.0,-9.0,-10.0,-15.0,-20.0];          % Powers (in dB)
K=[0,0,0,0,0,0];                                % Ricean Factor

Td=Td*(10^(-6));        % time delays (in sec)
nd=round(Td*Fs);        % time delay in samples
P=10.^(PdB/10);                 % Powers (Linear)
Np=length(xt);

yr=zeros(size(xt));

for k=1:length(Td)
g=jakes(Fd, Fs, Np);
s=sqrt(P(k)/(K(k)+1));                  % random path
m=sqrt(P(k)*K(k)/(K(k)+1));  % direct path
total=s*g+m*exp(j*2*pi*Fd/Fs*(0:length(xt)-1));
yr=yr+[zeros(1,nd(k)), xt(1:length(xt)-nd(k))].*total;
end
yr=reshape(yr,asn,asd).';
```

```
%channel_b
%-------------------------------------------------------------------------------
%
%Title : Severe Mobile channel
%Author : Prof.Roberto Cristi, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%-------------------------------------------------------------------------------
%Subroutines Used : jakes.m
%-------------------------------------------------------------------------------
function  yr=channel_b(xt)
[asd asn]=size(xt);
xt=xt.';
xt=xt(:).';
%
%  yr=channel(xt)
%  with xt=transmitted vector
%       yr=received vector
%       length(yr)=length(xt)
%
% if the program gives you an error, just increase the size of the
% transmitted vector "xt"

Fs=1.25*10^6;           % sampling frequency (Hz)
Fd=200;                 % doppler frequency (Hz)

% Parameters for Channel B
Td=[0.0, 0.25,9.0,13.0, 17.0, 20.0];    % time delays (in microsec)
PdB=[-2.5,0.0,-12.8,-10, -25.2, -16]; % Powers (in dB)
K=[0.5,0.5,0,0,0,0];                    % Ricean Factors

Td=Td*(10^(-6));        % time delays (in sec)
nd=round(Td*Fs);        % time delay in samples
P=10.^(PdB/10);                 % Powers (Linear)
Np=length(xt);

yr=zeros(size(xt));

for k=1:length(Td)
g=jakes(Fd, Fs, Np);
s=sqrt(P(k)/(K(k)+1));                  % random path
m=sqrt(P(k)*K(k)/(K(k)+1));  % direct path
total=s*g+m*exp(j*2*pi*Fd/Fs*(0:length(xt)-1));
yr=yr+[zeros(1,nd(k)), xt(1:length(xt)-nd(k))].*total;
end
yr=reshape(yr,asn,asd).';
```

```
%CHECK
%-------------------------------------------------------------------------------
%
%Title       : Source and Sink Message Checker
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
func-
tion[error_no,bit_error_total,freqerrs,errmx,rowerrs]=check(pic,x,xbit,y,ybit,n,k,blklgth)
if blklgth>n
disp('')
disp('ERROR! The block length(blklgth), must be equal to or less than the code word
length(n)')
disp('Please enter a smaller value for blklgth, or change n')
disp('')
elseif blklgth<=n
   if n<k
   disp('')
   disp('Error! The code word length(n) must be equal to or larger than the information
length(k)')
   disp('Please enter a larger value for n, or change k to a smaller number.')
   disp('')
   elseif n>=k
   First_matrix=x;
   Second_matrix=y;
   [rx cx]=size(x);
%
%Compare inputs x and y and generate error matrix, "errors"
   errors=(x~=y);
   First=xbit;
   Second=ybit;
   [rx1 cx1]=size(xbit);
%
%Compare inputs xbit and ybit and generate BIT error matrix, "bit_errors"
   bit_errors=(xbit~=ybit);
%
%Find the error distribution vs. OFDM frequencies
   freqerrs=sum(errors);
%
%Find the error location in "errors" where element in x and y differ.
   Error_locations=(find(errors))';
   Error_number=sum(sum(errors));
   Correct_symbl_num=(size(y,1)*size(y,2))-Error_number;
%
```

```
%Find the bit error location in "errors" where element in x1 and y1 differ.
%bit_Error_locations=(find(bit_errors))';
   bit_error_total=sum(sum(bit_errors));
%Correct_bit_num=(size(y1,1)*size(y1,2))-bit_Error_number;
%Reed-Solomon 8-bit symbol correction for (n-k)/2 symbols
   symcorr=floor((n-k)/2);
      if blklgth<=(n-k)
      disp('Error!!!The block length is too short for the given n and k values')
      disp('')
      elseif blklgth>(n-k)
      errtrans=errors';
%
%Reshape the error matrix as a vector of errors
      errvect=errtrans(:)';
      blkrem=rem(length(errvect),blklgth);
         if blkrem~=0;
         zeropad=zeros(blklgth-blkrem);
         errvectpad=[errvect zeropad(1,:)];
         elseif blkrem==0;
         errvectpad=errvect;
         end
         %
         blknos=length(errvectpad)/blklgth;
         errcorct=[];
         errblksum=[];
         %
         for lp=1:blknos;
         errblk=errvectpad(((blklgth*(lp-1))+1):(blklgth*lp));
         errblklgth=length(errblk);
            if sum(errblk)<=symcorr;
         noerr=zeros(errblklgth);
         errblk=noerr(1,:);
            elseif sum(errblk)>symcorr;
         errblk=errblk;
            end
         errcorct=[errcorct errblk];
         errblksum=[errblksum sum(errblk)];
         end
      newerrvect=errcorct(1:length(errvect));
      errtot=sum(newerrvect);
      RSerrs=(reshape(newerrvect,size(errors,2),size(errors,1)))';
%
%Find the error distribution vs. OFDM Frequencies
      freqerrs=sum(RSerrs);
      errindex=(find(RSerrs))';
```

```
        RSerrtot=sum(errblksum);
        RSerrdif=Error_number-RSerrtot;
        errperblk=[(1:blknos);errblksum];
%
%Check if x and y are the same. If not, display error message
        if x==y;
        disp('There are no errors!!!')
        error_no=0;
        errmx=errors;
        rowerrs=sum(errors');
        else
        disp('WARNING!:Errors were detected!')
        disp('')
        if n==k
        disp('WARNING!: Since n=k,there is no R-S error correcting possible')
        disp('')
        end
        disp(['For the given input parameters:n=',int2str(n),'and k=',int2str(k),',the Reed-
Solomon code is capable'])
        disp(['of correcting ',int2str(symcorr),'errors.'])
        disp('')
%
%Check if xbit and ybit are the same. If not, display error message
         if xbit==ybit;
        disp('There are no bit errors!!!')
        bit_error_no=0;
        bit_errmx=bit_errors;
        bit_rowerrs=sum(bit_errors');
        else
        disp('WARNING!:Errors were detected!')
        disp('')
            if n==k
             disp('WARNING!: Since n=k,there is no R-S error correcting possible')
             disp('')
            end
         end
%
%RS code was able to correct all errors
    if errtot==0
    Pre_RS_error_matrix=errors;
    disp('EXCELLENT: The Reed-Solomon code corrected all detected errors!')
    disp(['Originally the error total was:',int2str(Error_number)])
    disp('')
    error_no=0;
    errmx=zeros(rx,cx);
```

```
    rowerrs=sum(errmx');
%
%RS code was able to correct some errors but not all of them
    elseif errtot<Error_number
    Pre_RS_error_matrix=errors;
    Post_RS_error_matrix=RSerrs;
    errmx=RSerrs;
    rowerrs=sum(errmx');
    disp('The Reed-Solemon code corrected some detected errors, but not all.')
    disp(['Originally the error total was : ',int2str(Error_number)])
    disp('')
    disp(['After R-S decoding , the error number was reduced to:',int2str(RSerrtot)])
    disp('')
    error_no=RSerrtot;
    disp(['The total number of correct symbols are:',int2str((size(y,1)*size(y,2))-RSerrtot)])
    disp('')
    disp('The error number distribution per block number is :')
    disp(errperblk)
%
%RS code did not correct any errors
    elseif errtot==Error_number
    Error_matrix=errors;
    errmx=errors;
    rowerrs=sum(errors');
    disp('The Reed-Solomon code did not correct any errors.')
    disp('Perhaps a more powerful R-S code is required.')
    disp('')
    disp(['The total number of error occurrences is:',int2str(Error_number)])
    disp('')
    error_no=errtot;
    disp('The error number distribution per block number is :')
    disp(errperblk)
    end
end
end
end
end
disp('_____________________________________________________');
```

```
%CHUHF
%-----------------------------------------------------------------------------
%Title       : UHF Channel Model (multipath Channel Model 1)
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%-----------------------------------------------------------------------------
function y=chuhf(s,x,loss,dly,dop,N,freqspace)
c=10.^(-loss./20);
deltat=1/(N*freqspace);
d=(dly.*.000001)./deltat;
e=dop./freqspace;
[L,Nt]=size(x);
D=length(d);
x=x.';
x=x(:).';
%
%D path with delays from d.
xd=dline(x,d);
[rr,cc]=size(xd);
x=xd(1,:);
%
% Offsets direct path by .7 of max doppler freq.
xo=ofst(.7*e(1),N,x);
%
% First path with no fading.
for l=1:D
a=ray_dop(s,cc,N,e(1));
xd(1,:)=a.*xd(1,:);
end
%Sums the fading paths
y=c*xd;
%
%Adds in the First path without fading
y=y+xo;
y=y(1:L*Nt);
y=reshape(y,Nt,L).';
```

```
%CMV2FA
%-------------------------------------------------------------------------------
%
%Title       : Complex Frequency Array Generator
%Author      : Dave Roderick, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%Changes complex modulation values to frequency array
%-------------------------------------------------------------------------------
function X=cmv2fa(N,M)
[m n]=size(M);
if rem(n,2)==0;
M=M;
else
%
M=[zeros(m,1) M];
end
[m n]=size(M);
K=round(n/2);
%
%Generate a matrix of zeros with m row and N columns.
X=zeros(m,N);
%
X(:,1:K)=M(:,K+1:2*K);
X(:,N-K+1:N)=M(:,1:K);
```

```
%CNV_ENCD
%-------------------------------------------------------------------------------
%
%Title      : Convolutional Encoding
%Reference  : Contemporary Communication System using Matlab
%John G. Proakis & Masoud Salehi.
%-------------------------------------------------------------------------------
function ce_output=cnv_encd(ce_g,ce_k0,ce_input)
% cnv_encd(ce_g,ce_k0,ce_input)
% determines the output sequence of a binary convolutional encoder
% ce_g is the generator matrix of the convolutional code
% with ce_n0 rows and ce_l*ce_k0 columns. Its rows are ce_g1,ce_g2,....ce_gn.
% ce_k0 is the number of bits entering the encoder at each clock cycle.

% check to see if extra zero padding is necessary
if rem(length(ce_input),ce_k0)>0
ce_input=[ce_input,zeros(size(1:ce_k0-rem(length(ce_input),ce_k0)))];
end
ce_n=length(ce_input)/ce_k0;
%check the size of matrix ce_g
if rem(size(ce_g,2),ce_k0)>0
error('Error, ce_g is not of the right size.')
end
% determine ce_l and ce_n0
ce_l=size(ce_g,2)/ce_k0;
%disp(['The value of ce_l is:',int2str(ce_l)]);
ce_n0=size(ce_g,1);
%disp(' ')
%disp(['The value of ce_n0 is:',int2str(ce_n0)]);
%add extra zeros
ce_u=[zeros(size(1:(ce_l-1)*ce_k0)),ce_input,zeros(size(1:(ce_l-1)*ce_k0))];
%generate ce_uu, a matrix whose column are the contents of
%conv. encoder at various cycles.
ce_u1=ce_u(ce_l*ce_k0:-1:1);
for ce_i=1:ce_n+ce_l-2
ce_u1=[ce_u1,ce_u((ce_i+ce_l)*ce_k0:-1:ce_i*ce_k0+1)];
end
ce_uu=reshape(ce_u1,ce_l*ce_k0,ce_n+ce_l-1);
%determine the ce_output
ce_output=reshape(rem(ce_g*ce_uu,2),1,ce_n0*(ce_l+ce_n-1));
```

```
clc,close all
%COFDMSIM
%--------------------------------------------------------------------------------
%
%Title      : Simulation Of COFDM
%Author     : Dave Roderick, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
disp('_____________________________________________________________________');
disp('This batch m-file runs COFDM simulations using different channel models.')
%fort=input('To run the frequency version, enter 1(one), To run the time version, enter
0(zero), or to run both enter 2(two):');
fort=1; %frequency version
%freqno=input('Enter the # of OFDM frequencies (note : must be even):');
freqno=48;
%%%%N=input('Enter the number of FFT points (Note : This number must be larger
than # of OFDM frequencies):');
N=64;
chnmdl=input('Choose the channel model; 0-(Noise Free), 1-(AWGN), 21-(Mobile
Channel-1), 22-(MC-2), 23-(MC-3), 24-(MC-4)? :');
if chnmdl==0
   disp('Code Check simulation .');
   sigs=0;loss=0;dop=0;dly=0;
elseif chnmdl==1
   disp('AWGN Channel simulation .');
   sigs=input('Enter the noise variance(sigma) range or single value. (Ex lin-
space(0,0.02,20)or .003):');
   loss=0;dop=0;dly=0;
elseif chnmdl==21
   disp('Mobile Channel-1.');
   sigs=input('Enter the noise variance(sigma) range or single value. (Ex lin-
space(0,0.02,20)or .003):');

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

elseif chnmdl==22
   disp('Mobile Channel-2.');
   sigs=input('Enter the noise variance(sigma) range or single value. (Ex lin-
space(0,0.02,20)or .003):');
```

```
loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
   dop=[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

elseif chnmdl==23
   disp('Mobile Channel-3.');
   sigs=input('Enter the noise variance(sigma) range or single value. (Ex lin-
space(0,0.02,20)or .003):');
   loss=0;dop=0;dly=0;
elseif chnmdl==24
   disp('Mobile Channel-4.');
   sigs=input('Enter the noise variance(sigma) range (Ex: linspace(0,0.02,20)or .003):');
   loss=0;dop=0;dly=0;
end
%%%% Choosing Interleaver  %%%%
%allcase=input('Simulate all interleaver cases (yes) or specific ones(no)? (1=yes,0=no):');
allcase=0;
if allcase==1
   disp('All cases,(0-8),will be tested.');
   cases=[0:8];
elseif allcase==0
%%%%cases=input('Enter specific case numbers from (0 to 8)(Ex [0 4 5 8]):');
   cases=0; % Block interleaving chosen
end

if fort~=2
   if length(cases)~=1
      casey_n=input('Do you want to find optimal interleaver case(s) ? (1=yes, 0=no):');
   else
      casey_n=0;
   end
end
%%%%%
totsym=input('Enter the total minimum number of symbols to simulate (Ex 10000):');
rowno=ceil(totsym/freqno);
if totsym~=(rowno*freqno)
   disp(['Note:The actual total number of symbol to be simulated will be
:',int2str(rowno*freqno)]);
end
%pry_n=input('For the interleaver, do you want to calculate all possible intermediate ma-
trix dimension pairs?(1=yes,0=no):');
```

```
pry_n=1;
pair1=1;

pair2=rowno*freqno;
if pry_n==1
   Intrlvr_pairs=intlvprs(rowno,freqno);
   intlvrprs=Intrlvr_pairs;
   %%%% Desired interleaver pairs can be entered %%%%
   %disp('')
   disp('For these input parameters, all possible interleaver dimension pairs are: ')
   disp(Intrlvr_pairs)
end
pairs=input(['Desired interleaver pair? (Ex [row # col #] = [20 50] (Note: enter-
ing[',int2str(pair1),' ',int2str(pair2),'],or [',int2str(pair2),' ',int2str(pair1),'], offers no inter-
leaving functionality):']);
%rintlv=intlvrprs(8,1);cintlv=intlvrprs(8,2)
rintlv=pairs(1);
cintlv=pairs(2);

%mary=input('Enter the number of M-ary bits, q (i.e. for 256-ary, q=8):');
mary=1;
%nary=input('Enter the number of N-ary bits,q(i.e. for 16-ary, q=4):');
nary=2; %   QPSK is chosen
%freqspace=round(16600000/freqno);
freqspace=312500; % According to the standard 0.3125 Mhz of Frequency spacing.
%Ng=input('Enter the guard interval length (Number of sample points):');
Ng=16;
%ecc=input('Do you want to include Reed Solomon error correction coding ? (1=yes,
0=no):');
ecc=0;% Reed Solomon Error correction coding is not chosen
if ecc==1
   %code=input('Enter n,k and error correction block length (Ex [240 200 240]):');
   code=[240 200 240];
   n=code(1);
   k=code(2);
   blklgth=code(3);
elseif ecc==0
   n=freqno;
   k=freqno;
   blklgth=freqno;
end
svals=input('Enter specific seed values, or 0 for a random seed (ex [103 22, 60] or [0]):');
wait=3;pic=0;
svect=[];
for run=1:length(svals);
```

```
   errvect=[];
   errcase=[];
   errtot=[];
   if min(svals)==0
      rand('seed',sum(100*clock));
      s=round(abs(rand(1)*pi*10*(pic+1)*run));
   elseif min(svals)~=0
      s=svals(run);
   end
   svect=[svect,s];
   for l=1:length(cases);
   disp('_____________________________________________________________')
   disp(['Run#:',int2str(run)]);
   disp(['Seed=',int2str(s)]);
   disp(['Interleaver case=',int2str(cases(l))]);
      if fort<=1

chancdl(chnmdl,wait,pic,cases(l),s,freqno,rintlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,lo
ss,dly,dop,freqspace,fort);
      elseif fort==2
      disp('Frequency differential encoding/decoding simulation...')
      disp('_____________________________________________________________')

chancdl(chnmdl,wait,pic,cases(l),s,freqno,rintlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,lo
ss,dly,dop,freqspace,1);
      disp('************************************************************')
      disp('Time differential encoding/decoding simulation....')
      disp('_____________________________________________________________')

chancdl(chnmdl,wait,pic+12,cases(l),s,freqno,rintlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sig
s,loss,dly,dop,freqspace,0);
      end
   end
end
disp('*************************************************************************'
)
disp('');disp('Simulation finished!')
Seed=svect;
```

```
%CVDD
%-------------------------------------------------------------------------------
%
%Title      : Continuous Variable digital delay element.
%Reference  : C.W. Farrow, " A Continuously Variable Digital Element", IEEE
%International Symposium on Circuits & Systems,pp.2641-2645,1988.
%
%-------------------------------------------------------------------------------
function [y]=cvdd(x,alpha)
if ((nargin~=2)|(nargout~=1))
error('ERROR:usage:y=y=cvdd(x,alpha);');
return;
end
if (size(x)~=size(alpha))
error('ERROR:x and alpha must be the same size');
return;
end
if (abs(alpha)>0.5)
error('ERROR:alpha must be within -0.5 and 0.5');
return;
end
%
%-----------------------------------------------------
% Initialization
%-----------------------------------------------------
%
% Initialize FIR filter coefficients are in [1] (0,0.328 pass band)
C0=[-0.013824 0.054062 -0.157959 0.616394 0.616394 -0.157959 0.054062 -0.013824];
C1=[0.003143 -0.019287 0.1008 -1.226364 1.226364 -0.1008 0.019287 -0.003143];
C2=[0.055298 -0.216248 0.631836 -0.465576 -0.465576 0.631836 -0.216248 0.055298];
C3=[-0.012573 0.077148 -0.403198 0.905457 -0.905457 0.403198 -0.077148 0.012573];
%
%-----------------------------------------------------
% 4 parallel FIR and add together based on [1]
%-----------------------------------------------------
y0=filter(C0,[1],x);
y1=filter(C1,[1],x);
y2=filter(C2,[1],x);
y3=filter(C3,[1],x);
%
y=alpha.*y3;
y=alpha.*(y+y2);
y=alpha.*(y+y1);
y=y+y0;
```

```
%DECDRCDL
%-------------------------------------------------------------------------------
%
%Title       : COFDM Decoder With Deinterleaveing
%Author      : Dave Roderick, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
func-
tion[outmsg,viterbi_output_bit,random_msg,random_bit,M,MM]=decdrcdl(pic,dcase,K,F
a,nsymno,freqno,rdintlv,cdintlv,mary,nary,fort,B_random)
%
M=fa2cma(K,Fa);
Cmplx_mod_vals=M;
%
naryp=nary;
[s,MM]=dfdcdrft(naryp,nary,M,fort);
[L,cc]=size(s);
strans=s';
svect=strans(:).';
corrs=svect(1:nsymno);
%
nsymno;
Br=bm(mary,mb(nary,corrs));
lengthBr=length(Br);
rmndr=rem(length(Br),freqno);
if rmndr==0;
Br=Br;
elseif rmndr~=0;
Br=Br(1:(lengthBr-rmndr));
end
rcvd=(reshape(Br,freqno,length(Br)/freqno))';
Rcvd_Intlv_Ary=rcvd;
%
[Br Bc]=size(rcvd);
SYNC=[];
sr=rcvd';
si=sr(:)';
sd=cdldlv(rdintlv,cdintlv,dcase,si,SYNC);
received=reshape(sd,Bc,Br)';
viter_G=[1 0 1 1 0 1 1;1 1 1 1 0 0 1];
viter_k=1;
binary_value=mb(mary,sd);
[viterbi_output,survivor_sta,cumul_metrix]=viterbi(viter_G,viter_k,binary_value);
mary_dec=bm(mary,viterbi_output);
```

```
viterbi_output_bit=viterbi_output;
%outmsg=reshape(sd,Bc,Br)';
%
random_bit=B_random;
random_msg=bm(mary,random_bit);
[Brow Bcol]=size(random_msg);
%
outmsg=reshape(mary_dec,Bcol,Brow)';
Sink_Msg=outmsg;
```

```
%DECI2BIN
%-------------------------------------------------------------------------------
%
%Title      : Decimal to Binary Converter
%Author     : Tan Kok Chye, Naval Postgraduate School
%-------------------------------------------------------------------------------
function y=deci2bin(x,l)
y=zeros(1,l);
vi=1;
while x>=0 & vi<=l
y(vi)=rem(x,2);
x=(x-y(vi))/2;
vi=vi+1;
end
y=y(l:-1:1);
```

```
%DFDCDRFT
%-------------------------------------------------------------------------------
%
%Title       : Complex Number Demodulator & Frequency/Time Differential Decoder
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function [s,M]=dfdcdrft(qp,q,MD,fort)
if fort==0 %Time Differential decoding
%
MD=MD';
[m n]=size(MD);
%
% Perform a looping routine to find the phase differences between adjacent values in the
% array,MD,and put these calculated values into array,M.
   for l=1:m
      for j=1:n-1
         M(l,j)=MD(l,j+1)*conj(MD(l,j));
      end
   end
%
%Transpose the array back to its original form
M=M';
%
% Calculate the number of M-ary symbols based upon the exponent qp,then use this
number
% to find the number of equally spaced phases in a unit circle.
N=2^qp;
dph=2*pi/N;
%
% Divide the phase arguments of elements in M, by the equal phases generated by dph.
phn=angle(M)./dph;
%
% Calculate the phase sector number by finding the remainders.
s=rem(round(phn)+N,N);
elseif fort==1 % Frequency Differential decoding
%
% Transpose the modulation array, and find the dimensions
   [m,n]=size(MD);
   MD=MD(:,2:n);
   [m n]=size(MD);
%
% Perform a looping routine to find the phase differences between
% adjacent values in the array, MD, and put these calculated values into array,M.
```

```
        for l=1:m
            for j=1:n-1
                M(l,j)=MD(l,j+1)*conj(MD(l,j));
            end
        end
N=2^qp;
dph=2*pi/N;
%
% Calculate the phase sector number by finding the remainders.
phn=angle(M)./dph;
s=rem(round(phn)+N,N);
end
```

```
%DIFCDRFT
%-------------------------------------------------------------------------------
%
%Title       : Complex Number Modulator & Frequency/Time Differential Encoder
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%-------------------------------------------------------------------------------
function MD=difcdrft(q,m,fort)
if fort==0 %Time differential encoding
% M-ary alphabet size
N=2^q;
% Determine the number of equal phases based upon the m-ary symbol length
dph=2*pi/N;
% Find the size of the input symbol matrix
[rr n]=size(m);
%
% Perform the time differential encoding of phase values by cumulative summation,
% down one column at a time across the entire matrix. This function generates a matrix.
   for k=1:n
      md=cumsum(m(:,k));
% Generate the complex numbers with corresponding phase values.
      MD(:,k)=exp(i*dph.*md);
   end
%
% Inject the reference row of ones (zero phase) at top of output matrix for
% differential encoding synchronization
MD=[ones(1,n); MD];
elseif fort==1 % Frequency Differential encoding
%
% M-ary alphabet size
N=2^q;
dph=2*pi/N;
% Find the size of the input symbol matrix
[rr n]=size(m);
%
md=cumsum(m');
md=md';
%
% Generate the complex numbers with corresponding phase values.
MD=exp(i*dph.*md);
%
% Inject the reference row of ones (zero phase) at top of output matrix for
% differential encoding synchronization.
MD=[ones(rr,2) MD];
end
```

```
%DIFFCHKR
%-------------------------------------------------------------------------------
%
%Title       : Differential Encoder/Decoder Checker
%Author      : Dave Roderick, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function diffchkr(s,symno,freqno,mary,nary)
fort=input('For the frequency version, enter 1 (one); for the time version, enter 0 (zero):');
%
B=marymsg(mary,s,symno,freqno);
Rndm_m_ary_msg=B;
%
m1=bm(nary,mb(mary,B));
lengthm1=length(m1);
m=(reshape(m1,lengthm1/symno,symno))';
N_ary_msg=m;
%
if fort==1
disp('');disp('Frequency Differential Encoding/Decoding version')
%
%Freq. Diff. Enc.
%
MDD=difcdrf(mary,m);
elseif fort~=1
disp('');disp('Time Differential Encoding/Decoding version')
%
MDD=difcdrt(mary,m);
end
%
maryq=mary;
if fort==1
%
[s M]=difdcdrf(maryq,mary,MDD);
elseif fort ~=1
%
[s M]=difdcdrt(maryq,mary,MDD);
end
%
%Check results for correctness.
[error_no,freqerrs,errmx,rowerrs]=check(0,m,s,freqno,freqno,freqno);
```

```
%DLINE
%-------------------------------------------------------------------------------
%
%Title       : UHF Channel Delay Line Generator
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%-------------------------------------------------------------------------------
function xd=dline(x,d)
x=x.';
dmax=max(d);
dmin=min(d);
nmin=floor(dmin);
nmax=ceil(dmax);
x=[x;zeros(nmax+3,1)];
N=length(x);
Nd=length(d);
%
for n=1:Nd;
di=d(n);
D=floor(di);
deld=di-D;
xd(:,n)=cvdd(x,deld-.5);
xd(:,n)=[zeros(D,1);xd(1:N-D,n)];
end
xd=xd.';
[rr,cc]=size(xd);
xd=xd(:,4+nmin:cc);
```

```
%FA2CMA
%-------------------------------------------------------------------------------
%
%Title       : Frequency Array To Complex Modulation Array Converter
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function Mm=fa2cma(K,X)
[m n]=size(X);
Mm(:,1:K)=X(:,n-K+1:n);
Mm(:,K+1:2*K)=X(:,1:K);
Cmplx_mod_vals=Mm;
```

```
%FROFST
%-------------------------------------------------------------------------------
%Title  : Frequency Offset
%Author : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function [xofstd]=frofst(x,y)
[sati sutu]=size(x);
    x=x.';
    x=x(:).';
    wo=length(x);
    wu=1:wo;
    yd=wu*y
    xofst=x.*exp(i*(2.5*pi).*yd);
    xofst=xofst(1:sutu*sati);
    xofst=reshape(xofst,sutu,sati).';
    xofstd=xofst;
```

```
%INTLVCHK
%-------------------------------------------------------------------------------
%
%Title       : Interleaver/Deinterleaver Verifier
%Author      : Dave Roderick, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function intlvchk(s,symno,freqno,rintlv,cintlv,mary,dcase)
multiples=mltpl(symno,freqno);
Intrlvr_nbr_mltpls=multiples;
%
if (symno*freqno)~=(rintlv*cintlv)
disp('ERROR: The interleaver matrix dimensions are not compatible with the message
array size.')
disp(' Possible matrix dimensions are:');disp('')
disp(multiples)
disp('Note: The selected matrix dimensions can not accomodate the message array.')
disp(' In this case the number of rows times the number of columns is:');disp('')
disp(symno*freqno)
elseif(symno*freqno)/(rintlv*cintlv)==1
%
B=marymsg(mary,s,symno,freqno);
Random_msg=B
%
SYNC=[];
[Br Bc]=size(B);
Bt=B';
Bvect=Bt(:)';
si=cdlilv(rintlv,cintlv,dcase,Bvect,SYNC);
Bi=reshape(si,Bc,Br)';
Interleaved_array=Bi
%
[Br Bc]=size(Bi);
SYNC=[];
sr=Bi';
si=sr(:)';
sd=cdldlv(rintlv,cintlv,dcase,si,SYNC);
Bd=reshape(sd,Bc,Br)';
Deinterleaved_array=Bd
%
[error_no,freqerrs,errmx,rowerrs]=check(0,B,Bd,freqno,freqno,freqno);
end
```

```
%INTLVPRS
%-------------------------------------------------------------------------------
%
%Title       : Intermediate Matrix Interleaver Dimension Pairs
%Author      : Dave Roderick, Naval Postgraduate School
%Modified By : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function pairs=intlvprs(n,m)
prod=n*m;
multvect=[1];
for i=2:prod;
   remdr=rem(prod,i);
   if remdr==0
      multvect=[multvect i];
   else
   multvect=multvect;
   end
   mult=multvect;
end
lngth=length(mult);
nbr=mult(lngth);
result=[1 nbr];
for i=2:lngth;
crntpr=[mult(i) nbr/mult(i)];
result=[result;crntpr];
end
pairs=result;
```

```
%ITDA
%-------------------------------------------------------------------------------
%
%Title       : Frequency Domain Samples Without Guard Interval
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function Y=itda(Ng,y)
%
[L Nt]=size(y);
% Remove the guard interval for channel compensation.
%
y=y(:,Ng+1:Nt);
% Take the FFT of array, y
%
Y=fft(y.').';
```

```
%JAKES
%--------------------------------------------------------------------------------
%
%Title      : Multi-path
%Author     : Prof. Roberto Cristi, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
function g=jakes(Fd,Fs,N);
% g=jakes(Fd,Fs,N)
% generate a random vector of length N with jakes spectrum
% Fd=doppler frequency
% Fs=sampling frequency (Fs>10*Fd required)
% N=vector length
% g=generated random vector
% The vector g has unit power, ie. g'*g*Fs/N=1


L=round(log2(Fs/(10*Fd)));
F0=Fs/(2^L);            % sampling frequency for generating the random sequence
                        %
N0=2*ceil(N/(2^L));
Nf=512;                         % number of frequency components in spectrum definition
Nh=256;                         % length of FIR filter

% FIR Filter impulse response
w=0.54-0.46*cos(2*pi*(0:Nh-1)/Nh);          % hamming window

fd=Fd/F0;       % digital doppler frequency
f=0:(1/Nf):1-(1/Nf);    % vector of digital frequencies (1/2=Nyquist Freq.)
kd=floor(fd*Nf);        % index for doppler frequency fd

H0(1:kd)=sqrt(1-(f(1:kd)/fd).^2);
I=find(H0==0);  H0(I)=0.0001*ones(size(I));
H=zeros(1,Nf);
H(1:length(H0))=(1./H0);
H=H.*exp(-j*2*pi*f*Nh/2);
h0=real(ifft(H));
h=h0(1:length(w)).*w;
end

% Generate time varying taps at low sampling freq  Fs_ch
seed=14;
randn('state',seed);
a=randn(1,N0+Nh);
```

```
seed=seed+6;
b=randn(1,N0+Nh);
%x=randn(1,N0+Nh)+j*randn(1,N0+Nh);
x=a+j*b;

g=filter(h,1,x);
g=g(Nh+1:Nh+N0);     % steady state response


% Upsample to Fs=(2^L)*F0 in L stages
for m=1:L
F0=2*F0;
g=reshape([g;zeros(size(g))], 1,2*length(g));
omegad=2*pi*Fd/F0;
Domega=(pi/2)-omegad;
omegac=((pi/2)+omegad)/2;
M=ceil(((8*pi/Domega)-1)/2);
Nfilt=2*M+1;                                    % filter order using hamming window

nt=0:Nfilt-1;
w=0.54-0.46*cos(2*pi*nt/Nfilt);          % hamming window
hm(1:M)=sin(omegac*(nt(1:M)-M))./(pi*(nt(1:M)-M));
hm(M+1)=omegac/pi;
hm(M+2:Nfilt)=sin(omegac*(nt(M+2:Nfilt)-M))./(pi*(nt(M+2:Nfilt)-M));
hm=hm(1:Nfilt).*w;
g=filter(hm,1,g);
g=g(Nfilt:length(g));       % steady state
end

% Normalize
g=g(1:N);
K=g*g'/N;
g=g/sqrt(K);
```

```
%MARYMSG
%-------------------------------------------------------------------------------
%
%Title       : M-ary Message Test Pattern Generator
%Author      : Tan Kok Chye, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function [vmary_ce,random_bit]=marymsg(q,n,m)
%
[random_bit]=msg(n*m*q*.5-6);

conv_g=[1 0 1 1 0 1 1;1 1 1 1 0 0 1];
conv_k0=1;
conv_output=cnv_encd(conv_g,conv_k0,random_bit);
decml=bm(q,conv_output);
deciml=decml(1,1:(n*m));
vmary_ce=(reshape(deciml,m,n))';
```

```
%MB
%-------------------------------------------------------------------------------
%
%Title       : M-ary To Binary Converter
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%-------------------------------------------------------------------------------
function [b]=mb(q,m)
%
row=size(m,1);
col=size(m,2);
%
m=reshape(m',1,(row*col));
%
b0=rem(m,2);
m=(m-b0)./2;
B=b0;
%
for j=1:q-1
bj=rem(m,2);
m=(m-bj)./2;
B=[B;bj];
end
%
b=B(:)';
binary=b;
```

```
%METRIC
%-------------------------------------------------------------------------------
%
%Title       : Viterbi Hard Decision Decoding metric
%Author      : Tan Kok Chye, Naval Postgraduate School
%-------------------------------------------------------------------------------
function distance=metric(v_x,v_y)
if v_x==v_y
distance=0;
else
distance=1;
end
```

```
%MLTPL
%-------------------------------------------------------------------------------
%
%Title       : Common Multiples
%Author      : Tan Kok Chye, Naval Postgraduate School
%-------------------------------------------------------------------------------
function [mult]=mltpl(n,m)
max=n*m;
multvect=[1];
%
for i=2:max;
remdr=rem(max,i);
   if remdr==0
   multvect=[multvect i];
   else
   multvect=multvect;
   end
mult=multvect;
end
```

```
%MSG
%-------------------------------------------------------------------------------
%
%Title       : Message Test Pattern Generator
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function u=msg(k)
%
%rand('uniform');
%
%temp=rand('seed');
%
%rand('seed',s);
%
u=randint(1,k);
```

```
%NXT_STAT
%-------------------------------------------------------------------------------
%
%Title       : Next State
%Author      : Tan Kok Chye, Naval Postgraduate School
%-------------------------------------------------------------------------------
function [next_state,memory_contents]=nxt_stat(current_state,input,v_L,v_k)
binary_state=deci2bin(current_state,v_k*(v_L-1));
binary_input=deci2bin(input,v_k);
next_state_binary=[binary_input,binary_state(1:(v_L-2)*v_k)];
next_state=bin2deci(next_state_binary);
memory_contents=[binary_input,binary_state];
```

```
%OFST
%-------------------------------------------------------------------------------
%
%Title       : Channel Offset
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function xo=ofst(e,N,x)
[m Nt]=size(x);
xo=x.';
x=x.';
x=x(:);
x=x.';
Nt=length(x);
l=1:Nt;
%Creating the offset frequency
%
ex=x.*exp(i*(2*pi/N)*e.*l);
xo(:)=x;
xo=xo.';
```

```
%phsns2
%-------------------------------------------------------------------------------
%
%Title       : Phase Noise
%Author      : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%This m.file creates phase noise as a Wiener Process for all time samples
%-------------------------------------------------------------------------------
function phsnsig=phsns2(xmtifft,freqspace,perctB)
perctB;
[str stn]=size(xmtifft);
T =str*stn;
B=freqspace*perctB;
randn('state',100)  ;
dW = zeros(1,stn*str);
W = zeros(1,stn*str);
%
dW(1) = normrnd(0,sqrt(2*B));
W(1) = dW(1);
    for j = 2:(stn*str)
     dW(j) = normrnd(0,sqrt(2*B*j));
     W(j) = (W(j-1) + dW(j));
    end

    xmtifft=xmtifft.';
    xmtifft=xmtifft(:).';
    xmtifftpn=xmtifft.*exp(i*(2*pi*4*10^(-6)).*W);

    xmtifftpn=reshape(xmtifftpn,stn,str).';
    phsnsig=xmtifftpn;
    % phase noise plot
    figure(14)
    plot([0:T],[0,W],'r-')
    xlabel('t')
    ylabel('PN(t)')
    title('Phase Noise for all Time Domain Samples')
```

```
%phsns
%-------------------------------------------------------------------------------
%Title       : Phase Noise
%Author      : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%This m.file creates phase noise as a Wiener Process for each OFDM symbol
%-------------------------------------------------------------------------------
function phsnsig=phsns(xmtifft,freqspace,perctB)
[str stn]=size(xmtifft);
T =stn;
B=freqspace*perctB;

for i=1:str
   randn('state',100)  ;
   dW = zeros(1,stn);
   W = zeros(1,stn);
   dW(1) =normrnd(0,sqrt(2*B));
   W(1) = dW(1);
   for j = 2:stn
    dW(j) = normrnd(0,sqrt(2*B*j));
    W(j) = (W(j-1) + dW(j));
   end
   xmtifftpn(i,:)=xmtifft(i,:).*exp(i*(2*pi*4*10^(-6)).*W);
end
   phsnsig=xmtifftpn;
   % phase noise plot
   figure(14)
   plot([0:T],[0,W],'r-')
   xlabel('t')
   ylabel('PN(t)')
   title('Phase Noise Created for Each OFDM Symbol')
```

```
%RAY_DOP
%-------------------------------------------------------------------------------
%
%Title       : Rayleigh Doppler
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function c=ray_dop(s,M,N,es)
m=0:M-1;
randn('seed',s+10);
pr1=randn(1,20);
randn('seed',s+20);
pim=i*randn(1,20);
p=pr1+pim;
p=p/(40^.5);
rand('seed',s+30);
e=rand(1,20);
e=es*cos(2*pi*(e-.5));
E=exp(i*2*pi*e'*m/N);
c=p*E;
```

```
%ROTM
%-------------------------------------------------------------------------------
%
%Title       : Rotate Vector
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%-------------------------------------------------------------------------------
function [vp,vn]=rotm(v,m)
L=length(v);
m=rem(m,L);
ii=(1:L)-1;
isp=rem(ii-m+L,L)+1;
isn=rem(ii+m+L,L)+1;
vp=v(isp);
vn=v(isn);
```

```
%TDA
%-------------------------------------------------------------------------------
%
%Title       : Time Domain Samples With Guard Interval Precursor
%Author      : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by : Ahmet Yasin ERDOGAN, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
function x=tda(Ng,X)
[m N]=size(X);
%
% Perform inverse FFT on frequency values in array,X
x=ifft(X.');
% Add precursor of Ng samples to the beginning of the time domain array for channel
% compensation.
%
x=x.';
if Ng==0
x=x;
else
x=[x(:,N-Ng+1:N) x];
end
```

```
%VITERBI
%-------------------------------------------------------------------------------
%
%Title     : Viterbi Decoder
%Reference : Contemporary Communication System using MatLab
%John G. Proakis & Masoud Salehi.
%-------------------------------------------------------------------------------
func-
tion[decoder_output,survivor_state,cumulated_metric]=viterbi(v_G,v_k,channel_output)
%
v_n=size(v_G,1);
% check the sizes
if rem(size(v_G,2),v_k)~=0
error('Size of v_G and v_k do not agree')
end
%
if rem(size(channel_output,2),v_n)~=0
error('channel output not of the right size')
end
v_L=size(v_G,2)/v_k;
number_of_states=2^((v_L-1)*v_k);
%generate state transition matrix, output matrix, and input matrix
for v_j=0:number_of_states-1
   for v_l=0:2^v_k-1
   [next_state,memory_contents]=nxt_stat(v_j,v_l,v_L,v_k);
   input(v_j+1,next_state+1)=v_l;
   branch_output=rem(memory_contents*v_G',2);
   nextstate(v_j+1,v_l+1)=next_state;
   output(v_j+1,v_l+1)=bin2deci(branch_output);
   end
end
state_metric=zeros(number_of_states,2);
depth_of_trellis=length(channel_output)/v_n;
channel_output_matrix=reshape(channel_output,v_n,depth_of_trellis);
survivor_state=zeros(number_of_states,depth_of_trellis+1);
%start decoding of non-tail channel outputs
for v_i=1:depth_of_trellis-v_L+1
flag=zeros(1,number_of_states);
   if v_i<=v_L
   step=2^((v_L-v_i)*v_k);
   else
   step=1;
   end
   for v_j=0:step:number_of_states-1
      for v_l=0:2^v_k-1
```

```
        branch_metric=0;
        binary_output=deci2bin(output(v_j+1,v_l+1),v_n);
           for v_ll=1:v_n

branch_metric=branch_metric+metric(channel_output_matrix(v_ll,v_i),binary_output(v_
ll));
           end

if((state_metric(nextstate(v_j+1,v_l+1)+1,2)>state_metric(v_j+1,1)+branch_metric)|flag(
nextstate(v_j+1,v_l+1)+1)==0)
        state_metric(nextstate(v_j+1,v_l+1)+1,2)=state_metric(v_j+1,1)+branch_metric;
        survivor_state(nextstate(v_j+1,v_l+1)+1,v_i+1)=v_j;
        flag(nextstate(v_j+1,v_l+1)+1)=1;
        end
        end
    end
state_metric=state_metric(:,2:-1:1);
end
%start decoding of the tail channel_outputs
for v_i=depth_of_trellis-v_L+2:depth_of_trellis
flag=zeros(1,number_of_states);
last_stop=number_of_states/(2^((v_i-depth_of_trellis+v_L-2)*v_k));
    for v_j=0:last_stop-1
    branch_metric=0;
    binary_output=deci2bin(output(v_j+1,1),v_n);
       for v_ll=1:v_n

branch_metric=branch_metric+metric(channel_output_matrix(v_ll,v_i),binary_output(v_
ll));
        end

if((state_metric(nextstate(v_j+1,1)+1,2)>state_metric(v_j+1,1)+branch_metric)|flag(next
state(v_j+1,1)+1)==0)
        state_metric(nextstate(v_j+1,1)+1,2)=state_metric(v_j+1,1)+branch_metric;
        survivor_state(nextstate(v_j+1,1)+1,v_i+1)=v_j;
        flag(nextstate(v_j+1,1)+1)=1;
        end
    end
state_metric=state_metric(:,2:-1:1);
end
%generate the decoder output from the optimal path
state_sequence=zeros(1,depth_of_trellis+1);
state_sequence(1,depth_of_trellis)=survivor_state(1,depth_of_trellis+1);
for v_i=1:depth_of_trellis
```

```
state_sequence(1,depth_of_trellis-
v_i+1)=survivor_state((state_sequence(1,depth_of_trellis+2-v_i)+1),depth_of_trellis-
v_i+2);
end
decoder_output_matrix=zeros(v_k,depth_of_trellis-v_L+1);
for v_i=1:depth_of_trellis-v_L+1
dec_output_deci=input(state_sequence(1,v_i)+1,state_sequence(1,v_i+1)+1);
dec_output_bin=deci2bin(dec_output_deci,v_k);
decoder_output_matrix(:,v_i)=dec_output_bin(v_k:-1:1)';
end
decoder_output=reshape(decoder_output_matrix,1,v_k*(depth_of_trellis-v_L+1));
cumulated_metric=state_metric(1,1);
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. European Telecommunications Standards Institute (ETSI), *Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to Mobile, Portable and Fixed Receivers*, European Telecommunication Standard ETS 300 401, 1st edition, reference DE/JTC-DAB, February 1995. Available from the ETSI Secreteriat, F-06921 Sophia Antipolis Cedex, France.

2. European Telecommunications Standards Institute (ETSI), *Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television*, European Telecommunications Standard, ETS 300 744 1st edition, reference DE/JTC-DVB-8, March 1997. Available from the ETSI Secreteriat, F-06921 Sophia Antipolis Cedex, France.

3. J.J. van de Beek, P.A. Odling, S.K. Wilson, and P.O. Borjesson, "Orthogonal Frequency-Division Multiplexing (OFDM)", *Review of Radio Science 1996-1999*, W.R. Stone (ed.), International Union of Radio Science (URSI), Oxford University Press, Sweden, pp. 26-38, 1999.

4. T. Pollet, M.V. Bladel and M. Moeneclaey, "BER Sensitivity of OFDM systems to carrier frequency offset and Wiener phase noise," *IEEE Trans. on Commun.*, vol. 43, no. 2/3/4, pp. 191-193, Feb./Mar./Apr. 1995.

5. P.H. Moose, "A technique for orthogonal frequency division multiplexing frequency offset correction," *IEEE Trans. on Commun.*, vol. 42, no. 10, pp. 2908-2914, Oct. 1994.

6. A.G. Armada and M. Calvo, "Phase Noise and Sub-Carrier Spacing Effects on the Performance of an OFDM Communication System," *IEEE Commun Letters*, vol. 2, no. 1, January 1998.

7. Fuqin Xiong "The Effect of Doppler Frequency Shift, Frequency Offset of the Local Oscillators, and Phase Noise on the Performance of Coherent OFDM Receivers," prepared by Ohio Monty Andro Glenn Research Center, Cleveland, Ohio, NASA/TM—2001-210595, 2001.

8. Y. Zhao and S.G. Haggman, "Sensitivity to Doppler shift and carrier frequency errors in OFDM systems-The consequences and solutions", *Proceedings of IEEE, Vehicular Technology Conference*, pp. 2474-2478, 1996.

9. A.C. Brooks and S.J. Hoelzer, "Design and Simulation of Orthogonal Frequency Division Multiplexing," Final Report, Bradley University, Peoria, Illinois, 2001.

10. Chi-han Kao, "Performance of the IEEE 802.11a Wireless LAN Standard Over Frequency-Selective, Slow, Ricean Fading Channels," Master's thesis, Naval Postgraduate School, Monterey, California, 2002.

11. Institute of Electrical and Electronics Engineers, 802.11a, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer Extension in the 5 GHz Band*, 16 September 1999. http://ieeexplore.ieee.org, last accessed 26 February 2003.

12. The International Engineering Consortium, Web Forum Tutorials, *OFDM for Mobile Data Communication*, http://www.iec.org/, last accessed December 2003.

13. Klaus Witrisal, "Orthogonal Frequency Division Multiplexing (OFDM) for Broadband Communications and Digital Audio Broadcasting (DAB)" Presented in Graz University of Technology, Graz, Austria, 21 March 2002, http://spsc.inw.tugraz.at/courses/dat2/OFDM_handout.pdf, last accessed 26 February 2004.

14. Charan J. Langton, "Signal Processing & Simulation Newsletter", http://www.complextoreal.com/ISI.htm, last accessed December 2003.

15. K.T. Peh, "Orthogonal Frequency Division Multiplexing (OFDM) and Carrier Interferometry OFDM (CI/OFDM)" presented at Wireless Communications Seminar in Michigan Technological University, December 2003 FDM (CI/OFDM), http://www.ece.mtu.edu/faculty/ztian/ee5950/Teh_seminar.pdf, last accessed 26 February 2004.

16. Sven Vogeler, "Tutorial for HIPERLAN/2", University of Bremen, http://www.ant.uni-bremen.de/teaching/nsfzm/Tutorial_Hiperlan2.pdf, last accessed November 2003.

17. Bernard Sklar, *Digital Communications*, pp. 220-230, Prentice Hall, Upper Saddle River, New Jersey, 2001.

18. S.B. Weinstein and P.M. Ebert, "Data transmission by frequency-division multiplexing using the discrete Fourier transform," *IEEE Transactions on Communications*, vol. 19, no. 5, pp. 628-634, October 1971.

19. Serdar Umit Tezeren, "Reed-Muller codes in error correction in wireless ad-hoc networks," Master's thesis, Naval Postgraduate School, Monterey, California, 2004.

20. Richard Van Nee and Ramjee Prasad, *OFDM for Wireless Multimedia Communications*, pp. 73-92, The Artech House Universal Personal Communications, Norwood, 2000.

21. Juha Heiskala and John Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*, pp. 49-73, Sams Publishing, Indianapolis, 2002.

22. Mattias Olsson, "Rapid Prototype of an IEEE 802.11a Synchronizer," LITH-ISY-EX-3290-2002 Linkoping, 2002 http://www.ep.liu.se/exjobb/isy/2002/3290/exjobb.pdf, last accessed November 2003.

23. Theodore S. Rappaport, *Wireless Communications*, pp. 177-187, Prentice Hall, Upper Saddle River, New Jersey, 2002.

24. Daniel Landstrom, "Synchronization in OFDM systems," Licentiate in Engineering Thesis, Lund University, Lund Sweden, March 1999.

25. David V. Roderick, "A coded orthogonal frequency division multiplexing simulation of a high data rate, line-of-sight, digital radio for mobile maritime communications," Master's thesis, Naval Postgraduate School, Monterey, California, 1997.

26. Kok Chye Tan, "Development, simulation and evaluation of the IEEE 802.11a physical layer in a multi-path environment," Master's thesis, Naval Postgraduate School, Monterey, California, 2001.

27. Louis Thibault and Minh Thien Le. "Performance Evaluation of COFDM for Digital Audio Broadcasting. Part 1: Parametric study" *IEEE Trans. on Broadcasting*, vol. 43, no. 1, pp. 64-75, Mar. 1997.

28. R.C. North, W.D. Bryan, R.A. Axford, Jr., K.C. Owens, D.R. Butts, B. Watkins and P.D. Donich, "Use of the AN/WSC-3 External Modem Interface for High-Data-Rate UHF Digital Communication", Technical Report 1701, Naval Command, Control and Ocean Surveillance Center RDT&E Division, San Diego, CA, May 1995.

29. William C. Jakes, *Microwave Mobile Communications*, pp. 11-24, IEEE Press, Piscataway, New Jersey, 1993.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code EC
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Murali Tummala, Code EC/TU
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor Roberto Cristi, Code EC/CX
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

6. Dr. Richard North
   Deputy Technical Director
   PEO C4I and Space
   SPAWAR Systems Center San Diego
   San Diego, California

7. Ahmet Yasin ERDOGAN
   337 Sok. No.59 Daire 6
   Sirinyer, Izmir, TURKEY